Abstract

This document discusses scenarios and requirements for Autonomic Control Planes (ACPs) constructed and secured at Layer 2. These would be alternatives to an ACP constructed and secured at the network layer. A secure ACP is required as the substrate for the Generic Autonomic Signaling Protocol (GRASP) used by Autonomic Service Agents.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 1, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

As defined in [I-D.ietf-anima-reference-model], the Autonomic Service Agent (ASA) is the atomic entity of an autonomic function, and it is instantiated on autonomic nodes. When ASAs communicate with each other, they should use the Generic Autonomic Signaling Protocol (GRASP) [I-D.ietf-anima-grasp]. It is essential that such communication is strongly secured to avoid malicious interference with the Autonomic Infrastructure (ANI).

For this reason, GRASP must run over a secure substrate that is isolated from regular data plane traffic. This substrate is known as the Autonomic Control Plane (ACP). A method for constructing an ACP at the network layer is described in [I-D.ietf-anima-autonomic-control-plane]. The present document discusses scenarios and requirements for constructing an ACP at layer 2.

2. Network Scenarios Suitable for a Layer 2 ACP

The ANI design is aimed at managed networks, as explained in the reference model [I-D.ietf-anima-reference-model]. For a wide area network (such as a large campus, a multi-site enterprise network, or a carrier network considered as a whole) it is appropriate to construct the ACP using network layer techniques and network layer security. and that is the model described in [I-D.ietf-anima-autonomic-control-plane]. However, in at least two cases an ACP covering a smaller geographical area may be appropriate:
1. A small enterprise that is completely within one building or several adjacent buildings, but is large enough to require autonomic network management.

2. An enterprise that prefers in any case to segment its network into smaller units for management purposes.

In either case, we assume that the L2 ACP may extend into the Network Operations Centre (NOC) so that it can be interfaced to traditional tools for Operations, Administration and Maintenance, as described in [RFC8368]. In the terminology of that document, an L2 ACP is an instance of a Generalized ACP.

3. Requirements for a Layer 2 Technology

1. The technology must support transmission of IPv6 packets according to [RFC8200]. Since GRASP can run on a single network segment using link-local addresses, there is not required to be an IPv6 router or DHCPv6 server.

2. The technology must support multicast. If the switches are not completely transparent to layer 2 multicast, they must support Multicast Listener Discovery Version 2 (MLDv2) for IPv6 [RFC3810].

3. The technology should have a minimum MTU of 1500 bytes.

4. The technology must support isolation of a given set of nodes (the "ACP VLAN").

5. The technology must support secure authorization for access to the ACP VLAN. If the VLAN technology in use does not support password protection, a VLAN access control list could be used.

6. The technology should support both the normal dataplane VLAN and the ACP VLAN on the same physical sockets. (Possibly the dataplane may be the native VLAN, i.e. frames with no VLAN tag.)

7. The technology should support line speed encryption of the ACP VLAN.

8. The technology should support wired/wireless bridging if relevant.

9. The technology should require minimal manual configuration of ACP nodes. However, it is expected that the nodes will need to be preconfigured before deployment with the VLAN ID, and a password or encryption key if necessary. A solution which is both secure
and self-configuring at Layer 2 is out of scope for this document.

A small ACP software module will be needed in each autonomic node, whose job is to provide the GRASP core with the following information about the L2 ACP:

1. A signal that the L2 ACP is available and secure.

2. The current global scope IPv6 address that GRASP should use as its primary locator, preferably a ULA, if available. As mentioned, if no such address is available, GRASP will simply operate with link-local addresses.

3. A list of [interface_index, link_local_address] pairs for all valid IPv6 interfaces attached to the L2 ACP. The interface index is an integer for maximum portability between operating systems.

4. Multiple Segments

This section is for further study.

The L2 ACP could in principle be extended across multiple segments or even multiple sites by use of secure L2VPN technology.

5. Implementation Status [RFC Editor: please remove]

A simple ACP software module emulating that needed for a secure L2 ACP has been implemented, but it does not in fact verify security. It may be found at <https://github.com/becarpenter/graspy/blob/master/acp.py> and is briefly documented in <https://github.com/becarpenter/graspy/blob/master/graspy.pdf>.

6. Security Considerations

The assumption of this document is that any Layer 2 solution chosen must have adequate security against interlopers and eavesdroppers. It should be noted that (at least in a wired network) this also requires adequate physical security to prevent access by unauthorized persons, including physical intrusion detection.

The fact that an IPv6 router is not required in an L2 ACP excludes many Layer 3 vulnerabilities by construction. No outside entity can generate link-local IPv6 packets, and no outside entity can send global scope packets to any autonomic node.
7. IANA Considerations

This document makes no request of the IANA.

8. Acknowledgements

Excellent suggestions were made by TBD and other participants in the ANIMA WG.

9. References

9.1. Normative References


9.2. Informative References

[I-D.ietf-anima-autonomic-control-plane]

[I-D.ietf-anima-grasp]

[I-D.ietf-anima-reference-model]

Appendix A. Change log [RFC Editor: Please remove]

draft-carpenter-anima-l2acp-scenarios-00, 2019-02-28:

Initial version

Authors’ Addresses

Brian Carpenter
The University of Auckland
School of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Bing Liu (editor)
Huawei Technologies
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com
This document outlines the challenges associated with implementing Bootstrapping Remote Secure Key Infrastructures over IEEE 802.11 and IEEE 802.1x networks. Multiple options are presented for discovering and authenticating to the correct IEEE 802.11 SSID. This draft is a discussion document and no final recommendations are made on the recommended approaches to take. However, the advantages and downsides of each possible method are evaluated.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
   1.1. Terminology .............................................. 4
2. Discovery and Authentication Design Considerations ............. 5
   2.1. Incorrect SSID Discovery ................................. 5
       2.1.1. Leveraging BRSKI MASA .......................... 5
       2.1.2. Relying on the Network Administrator ............ 6
       2.1.3. Requiring the Network to Demonstrate Knowledge of Device ........................................ 6
   2.2. IEEE 802.11 Authentication Mechanisms .................. 7
       2.2.1. Authentication Signaling Considerations ........... 7
       2.2.2. IP Address Assignment Considerations ............. 8
   2.3. Client and Server Implementations ........................ 8
3. Potential SSID Discovery and Validation Mechanisms ............. 9
   3.1. Well-known BRSKI SSID .................................. 10
   3.2. IEEE 802.11aq ........................................... 11
   3.3. IEEE 802.11 Vendor Specific Information Element ....... 12
   3.4. Reusing Existing IEEE 802.11u Elements ................. 12
   3.5. IEEE 802.11u Interworking Information - Internet ........ 13
   3.6. Define New IEEE 802.11u Extensions ..................... 13
   3.7. Wi-Fi Protected Setup ................................... 14
   3.8. Define and Advertise a BRSKI-specific AKM in RSNE ....... 14
   3.9. Wi-Fi Device Provisioning Profile ...................... 15
4. Potential Mutual Validation Options ........................... 16
   4.1. MAC Address Validation method ........................... 16
   4.2. Vendor Token Validation method .......................... 16
   4.3. Device Token Validation method .......................... 17
   4.4. Infrastructure Response Filtering ....................... 17
   4.5. Infrastructure Validation Method ....................... 17
5. Potential Authentication Options ................................. 18
   5.1. Unauthenticated and Unencrypted or OWE Pre-BRSKI and EAP-TLS Post-BRSKI .................. 19
   5.2. DPP Pre-BRSKI and EAP-TLS Post-BRSKI .................... 19
   5.3. PSK or SAE Pre-BRSKI and EAP-TLS Post-BRSKI ................ 19
   5.4. MAC Address Bypass Pre-BRSKI and EAP-TLS Post-BRSKI ... 20
   5.5. EAP-TLS Pre-BRSKI and EAP-TLS Post-BRSKI ................ 20
   5.6. New DPP BRSKI mechanism ................................ 21
   5.7. New TEAP BRSKI mechanism ............................... 21
   5.8. New IEEE 802.11 Authentication Algorithm for BRSKI and EAP-TLS Post-BRSKI .................. 24
   5.9. New IEEE 802.1X EAPOL-Announcements to encapsulate BRSKI
1. Introduction

Bootstrapping Remote Secure Key Infrastructures (BRSKI) [I-D.ietf-anima-bootstrapping-keyinfra] describes how a device can bootstrap against a local network using an Initial Device Identity (IDI) X.509 [IEEE802.1AR] IDevID certificate that is pre-installed by the vendor on the device in order to obtain an [IEEE802.1AR] LDevID. The BRSKI flow assumes the device can obtain an IP address, and thus assumes the device has already connected to the local network. Further, the draft states that BRSKI use of IDevIDs:

allows for alignment with [IEEE802.1X] 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.

The draft does not describe any mechanisms for how an [IEEE802.11] enabled device would discover and select a suitable [IEEE802.11] SSID when multiple SSIDs are available. A typical deployment scenario could involve a device being deployed in a location where there are twenty or more SSIDs being broadcast, for example, in a multi-tenant building or campus where multiple independent organizations operate [IEEE802.11] networks.

In order to reduce the administrative overhead of installing new devices, it is desirable that the device will automatically discover and connect to the correct SSID without the installer having to manually provision any network information or credentials on the device. It is also desirable that the device does not discover, connect to, and automatically enroll with the wrong network as this could result in a device that is owned by one organization connecting to the network of a different organization in a multi-tenanted building or campus.
Additionally, as noted above, the BRSKI draft does not describe how BRSKI could potentially align with [IEEE802.1X] authentication mechanisms.

This document outlines multiple different potential mechanisms that would enable a bootstrapping device to choose between different available [IEEE802.11] SSIDs in order to associate and execute the BRSKI flow. This document also outlines several options for how [IEEE802.11] networks enforcing [IEEE802.1X] authentication could enable the BRSKI flow, and describes the required device behaviour.

This document presents both [IEEE802.11] mechanisms and Wi-Fi Alliance (WFA) mechanisms. An important consideration when determining what the most appropriate solution to device onboarding should be is what bodies need to be involved in standardisation efforts: IETF, IEEE and/or WFA.

1.1. Terminology

IEEE 802.11u: an amendment to the IEEE 802.11-2007 standard to add features that improve interworking with external networks.

ANI: Autonomic Networking Infrastructure

ANQP: Access Network Query Protocol

AP: IEEE 802.11 Access Point

CA: Certificate Authority

EAP: Extensible Authentication Protocol

EST: Enrollment over Secure Transport

HotSpot 2.0 / HS2.0: An element of the Wi-Fi Alliance Passpoint certification program that enables cell phones to automatically discover capabilities and enroll into IEEE 802.11 guest networks (hotspots).

IE: Information Element

IDevID: Initial Device Identifier

LDevID: Locally Significant Device Identifier

OI: Organization Identifier

MASA: BRSKI Manufacturer Authorized Signing Authority service
2. Discovery and Authentication Design Considerations

2.1. Incorrect SSID Discovery

As will be seen in the following sections, there are several discovery scenarios where the device can choose an incorrect SSID and attempt to join the wrong network. For example, the device is being deployed by one organization in a multi-tenant building, and chooses to connect to the SSID of a neighbor organization. The device is dependent upon either detecting that the other networks are unwanted candidates, or upon the incorrect networks rejecting its BRSKI enrollment attempt. It is possible that the device could end up enrolled with the wrong network. It is also possible that the device will waste time before identifying and joining the correct network.

2.1.1. Leveraging BRSKI MASA

2.1.1.1. Prevention

BRSKI allows optional sales channel integration which could be used to ensure only the "correct" network can claim the device. In theory, this could be achieved if the BRSKI MASA service has explicit knowledge of the network where every single device will be deployed. After connecting to the incorrect SSID and possibly authenticating to the network, the device would present network TLS information in its voucher-request, and the MASA server would have to reject the request based on this network TLS information and not issue a voucher. The device could then reject that SSID and attempt to bootstrap against the next available SSID.

This could possibly be achieved via sales channel integration, where devices are tracked through the supply chain all the way from manufacturer factory to target deployment network operator. In practice, this approach may be challenging to deploy as it may be extremely difficult to implement this tightly coupled sales channel
integration and ensure that the MASA actually has accurate deployment network information.

An alternative to sales channel integration is to provide the device owners with a, possibly authenticated, interface or API to the MASA service whereby they would have to explicitly claim devices prior to the MASA issuing vouchers for that device. There are similar problems with this approach, as there could be a complex sales and channel partner chain between the MASA service operator and the device operator who owns and deploys the device. This could make exposure of APIs by the MASA operator to the device operator untenable.

2.1.1.2. Detection

If a device connects to the wrong network, the correct network operator could detect this incorrect association after the fact by integration with MASA and checking audit logs for the device. The MASA audit logs should indicate all networks that have been issued vouchers for a specific device. This mechanism also relies on the correct network operator having a list, bill or materials, or similar of all device identities that should be connecting to their network in order to check MASA logs for devices that have not come online, but are known to be physically deployed.

2.1.2. Relying on the Network Administrator

An obvious mechanism is to rely on network administrators to be good citizens and explicitly reject devices that attempt to bootstrap against the wrong network. This is not guaranteed to work for two main reasons:

- Some network administrators will configure an open policy on their network. Any device that attempts to connect to the network will be automatically granted access.
- Some network administrators will be bad actors and will accept the onboarding of devices that they do not own but that are in range of their networks.

2.1.3. Requiring the Network to Demonstrate Knowledge of Device

Technologies such as the WFA Easy Connect (also known as Device Provisioning Profile [DPP]) require that a network provisioning entity demonstrates knowledge of device information such as the device's bootstrapping public key prior to the device attempting to connect to the network. This gives a higher level of confidence to the device that it is connecting to the correct SSID. These
mechanisms could leverage a key that is printed on the device label, or included in a sales channel bill of materials. The security of these types of key distribution mechanisms relies on keeping the device label or bill of materials content from being compromised prior to device installation.

[IEEE802.11] also includes several advertisement mechanisms that could allow the device to exchange information with the wireless infrastructure. Examples are provided throughout this text. Such exchange can be added to, or integrated with, the standard [IEEE802.11] discovery mechanisms to allow the device to discard the networks that would not provide information showing that the network knows the device. Similarly, the network could reject the association of devices that would fail to show particular indicators related to their credentials.

2.2. IEEE 802.11 Authentication Mechanisms

[IEEE802.11i] allows an SSID to advertise different authentication mechanisms via the AKM Suite list in the RSNE. A very brief introduction to [IEEE802.11i] is given in the appendices. An SSID could advertise PSK or [IEEE802.1X] authentication mechanisms. When a network operator needs to enforce two different authentication mechanisms, one for pre-BRSKI devices and one for post-BRSKI devices, the operator has four options:

- configure two SSIDs with the same SSID string value, each one advertising a different authentication mechanism
- configure two different SSIDs, each with its own SSID string value, with each one advertising a different authentication mechanism
- configure a single SSID, advertising two different authentication mechanisms in the RSNE
- configure a single SSID, advertising a general authentication mechanism in the RSNE, and particular additional authentication options in some other information element.

If devices have to be flexible enough to handle two or more of these options, then this adds complexity to the device firmware and internal state machines. Similarly, if network infrastructure (APs, WLCs, AAAs) potentially needs to support all options, then this adds complexity to network infrastructure configuration flexibility, software and state machines. Consideration must be given to the practicalities of implementation for both devices and network infrastructure when designing the final bootstrap mechanism and
2.2.1. Authentication Signaling Considerations

Devices should be flexible enough to handle potential options defined by any final draft. When discovering a pre-BRSKI SSID, the device should also discover the authentication mechanisms enforced by the SSID. If the device supports the authentication mechanism being advertised, then the device can connect to the SSID in order to initiate the BRSKI flow. For example, the device may support [IEEE802.1X] as a pre-BRSKI authentication mechanism, but may not support PSK as a pre-BRSKI authentication mechanism.

Once the device has completed the BRKSI flow and has obtained an LDevID, a mechanism is needed to tell the device which SSID to use for post-BRSKI network access. This may be the same SSID as the pre-BRSKI SSID, or another SSID. The decision in whether to onboard devices through the production SSID or use an onboarding and provisioning SSID that is different from the production SSID is dependent on individual organisation networking and security architectures. As such, the mechanism by which the post-BRSKI SSID is advertised to the device, if that SSID is different from the pre-BRSKI SSID, is out-of-scope of this version of this document.

2.2.2. IP Address Assignment Considerations

If a device has to perform two different authentications, one for pre-BRSKI and one for post-BRSKI, network policy will typically assign the device to different VLANs for these different stages, and may assign the device different IP addresses depending on which network segment the device is assigned to. This could be true even if a single SSID is used for both pre-BRSKI and post-BRSKI connections. Therefore, the bootstrapping device may need to completely reset its network connection and network software stack, and obtain a new IP address between pre-BRSKI and post-BRSKI connections.

2.3. Client and Server Implementations

When evaluating all possible SSID discovery mechanisms and authentication mechanisms outlined in this document, consideration must be given to the complexity of the required client and server implementation and state machines. Consideration must also be given to the network operator configuration complexity if multiple permutations and combinations of SSID discovery and network authentication mechanisms are possible.
3. Potential SSID Discovery and Validation Mechanisms

This section outlines multiple different mechanisms that could potentially be leveraged that would enable a bootstrapping device to choose between multiple different available [IEEE802.11] SSIDs. The discovery mechanism needs to include the following steps:

- A process for the bootstrapping device that has not completed the bootstrapping process, and that it is at a stage where such process is needed before further connection

- A process for the Wi-Fi infrastructure to signal that it can perform bootstrapping

- A process for the bootstrapping device and the infrastructure to validate each other request. This step includes, for the bootstrapping device, discriminating between two SSIDs in range. This step may also include, for the Wi-Fi infrastructure, validating the bootstrapping device’s request (before accepting it).

The discovery options outlined in this document include:

- Well-known BRSKI SSID
- [IEEE802.11aq]
- [IEEE802.11] Vendor Specific Information Element
- Reusing Existing [IEEE802.11u] Elements
- [IEEE802.11u] Interworking Information - Internet
- Define New [IEEE802.11u] Extensions
- Wi-Fi Protected Setup
- Define and Advertise a BRSKI-specific AKM in RSNE
- Wi-Fi Device Provisioning Profile

These mechanisms are described in more detail in the following sections.
3.1. Well-known BRSKI SSID

A standardized naming convention for SSIDs offering BRSKI services is defined such as:

- BRSKI%ssidname

Where:

- BRSKI: is a well-known prefix string of characters. This prefix string would be baked into device firmware.
- %: is a well known delimiter character. This delimiter character would be baked into device firmware.
- ssidname: is the freeform SSID name that the network operator defines.

Device manufacturers would bake the well-known prefix string and character delimiter into device firmware. Network operators configuring SSIDs which offer BRSKI services would have to ensure that the SSID of those networks begins with this prefix. On bootstrap, the device would scan all available SSIDs and look for ones with this given prefix.

If multiple SSIDs are available with this prefix, then the device could simply round robin through these SSIDs and attempt to start the BRSKI flow on each one in turn until it succeeds.

This mechanism suffers from the limitations outlined in Section 2.1 - it does nothing to prevent a device enrolling against an incorrect network.

Another issue with defining a specific naming convention for the SSID is that this may require network operators to have to deploy a new SSID. In general, network operators attempt to keep the number of unique SSIDs deployed to a minimum as each deployed SSID eats up a percentage of available air time and network capacity. A good discussion of SSID overhead and an SSID overhead [calculator] is available.

Additionally, a third issue with this mechanism is that the bootstrapping SSID might be different from the production SSID. As such, using this mechanism may force a network operator to maintain an SSID (with the overhead concerns detailed above) just for occasional bootstrapping events. The SSID could be enabled only when bootstrapping events are expected, but this manual operation does not scale very well (and ignores cases where devices need to re-bootstrap
or are introduced into the network individually at unpredictable intervals). Keeping the SSID enabled at all times consumes airtime for low added value outside of the bootstrapping events.

3.2. IEEE 802.11aq

[IEEE802.11aq] is an amendment to the [IEEE802.11] Standard that was published in August 2018. [IEEE802.11aq] defines new elements that can be included in [IEEE802.11] Beacon, Probe Request and Probe Response frames, and defines new elements for ANQP frames.

The extensions allow an AP to broadcast support for backend services, where allowed services are those registered in the [IANA] Service Name and Transport Protocol Port Number Registry. The services can be advertised in [IEEE802.11] elements that include either:

- SHA256 hashes of the registered service names
- a bloom filter of the SHA256 hashes of the registered service names

Bloom filters simply serve to reduce the size of Beacon and Probe Response frames when a large number of services are advertised. If a bloom filter is used by the AP, and a device discovers a potential service match in the bloom filter, then the device can query the AP for the full list of service name hashes using newly defined ANQP elements.

If BRSKI were to leverage [IEEE802.11aq], then a BRSKI service would need to be defined in [IANA].

[IEEE802.11aq] describes two types of exchanges. An unsolicited Preassociation Discovery (PAD) procedure, where the AP advertises services reachable through the AP, and a solicited method, where the PAD is initiated by the unassociated client attempting to discover a service offered through the AP and SSID. The unsolicited PAD method could be leveraged to advertise support for BRSKI. This mechanism suffers from the limitations outlined in Section 2.1 - it does nothing to prevent a device enrolling against an incorrect network.

The solicited method could be used by the device to query about general BRSKI support, or to request information about specific BRSKI modes or options. This method could be used to overcome the Section 2.1 issue.
3.3. IEEE 802.11 Vendor Specific Information Element

[IEEE802.11] defines Information Element (IE) number 221 for carrying Vendor Specific information. The purpose of this document is to define an SSID discovery mechanism that can be used across all devices and vendors, so use of this IE is not an appropriate long term solution.

3.4. Reusing Existing IEEE 802.11u Elements

[IEEE802.11u] defines mechanisms for interworking. An introduction to [IEEE802.11u] is given in the appendices. Existing IEs in [IEEE802.11u] include:

- Roaming Consortium IE (RCOI)
- NAI Realm IE

These existing IEs could be used to advertise a well-known, logical service that devices implicitly know to look for. This may be implemented in the spirit of the 802.11u logic, where the NAI or the RCOI point to a specific set of service providers. This could also be implemented as a variation where the NAI or the RCOI point to a specific service, with no specific service provider identified in the IE.

In the case of NAI Realm, a well-known service name such as ",_bootstrapks" could be defined and advertised in the NAI Realm IE. In the case of Roaming Consortium, a well-known Organization Identifier (OI) could be defined and advertised in the Roaming Consortium IE.

Device manufacturers would bake the well-known NAI Realm or Roaming Consortium OI into device firmware. Network operators configuring SSIDs which offer BRSKI services would have to ensure that the SSID offered this NAI Realm or OI. On bootstrap, the device would scan all available SSIDs and use ANQP to query for NAI Realms or Roaming Consortium OI looking for a match.

The key concept with this proposal is that BRSKI uses a well-known NAI Realm name or Roaming Consortium OI more as a logical service advertisement rather than as a backhaul internet provider advertisement. This is conceptually very similar to what [IEEE802.11aq] is attempting to achieve.

Leveraging NAI Realm or Roaming Consortium would not require any [IEEE802.11] specification changes, and could be defined by this IETF draft with the strings suggested above for NAI. However, the RCOI
has the format of a MAC address, and would need to be allocated by
the IEEE. In the case where specific vendors would implement a
specific NAI or RCOI, identifying both the vendor or vendor
consortium and support for BRSKI, new NAI and RCOI would need to be
defined by these vendors. Although the Wireless Broadband Alliance
(WBA) keeps a Next generation Hotspot (NGH) registry of known RCOIs
and NAIs, there is no official and exhaustive published repository of
these values.

In addition to BRSKI support, as the NAI Realm includes advertising
the EAP mechanism required, if a new EAP-BRSKI were to be defined,
then this could be advertised. Devices could then scan for an NAI
Realm that enforced EAP-BRSKI, and ignore the realm name.

This mechanism suffers from the limitations outlined in Section 2.1 -
it does nothing to prevent a device enrolling against an incorrect
network.

Additionally, as the IEEE is attempting to standardize logical
service advertisement via [IEEE802.11aq], [IEEE802.11aq] would seem
to be the more appropriate option than overloading an existing IE.
However, it is worth noting that configuration of 802.11u IEs is
commonly supported today by Wi-Fi infrastructure vendors, and this
mechanism may be suitable for demonstrations or proof-of-concepts.

3.5. IEEE 802.11u Interworking Information - Internet

It is possible that an SSID may be configured to provide unrestricted
and unauthenticated internet access. This could be advertised in the
Interworking Information IE by including:

- internet bit = 1
- ASRA bit = 0

If such a network were discovered, a device could attempt to use the
BRSKI well-known vendor cloud Registrar. Possibly this could be a
default fall back mechanism that a device could use when determining
which SSID to use. However, this mechanism suffers from the
limitations outlined in Section 2.1 - it does nothing to prevent a
device enrolling against an incorrect network. Additionally, this
mechanism does not provide any information about local BRSKI support.

3.6. Define New IEEE 802.11u Extensions

Of the various elements currently defined by [IEEE802.11u] for
potentially advertising BRSKI, NAI Realm and Roaming Consortium IE
are the two existing options that are a closest fit, as outlined
above. Another possibility that has been suggested in the IETF mailers is defining an extension to [IEEE802.11u] specifically for advertising BRSKI service capability. Any extensions should be included in Beacon and Probe Response frames so that devices can discover BRSKI capability without the additional overhead of having to explicitly query using ANQP. ANQP queries could be used to provide additional information, such as vendor support.

[IEEE802.11aq] appears to be the proposed mechanism for generically advertising any service capability, provided that service is registered with [IANA]. It is probably a better approach to encourage adoption of [IEEE802.11aq] and register a service name for BRSKI with [IANA] rather than attempt to define a completely new BRSKI-specific [IEEE802.11u] extension.

3.7. Wi-Fi Protected Setup

Wi-Fi Protected Setup (WPS) only works with Wi-Fi Protected Access (WPA) and WPA2 when in Personal Mode. WPS does not work when the network is in Enterprise Mode enforcing [IEEE802.1X] authentication. WPS is intended for consumer networks and does not address the security requirements of enterprise or IoT deployments. Additionally, WPS relies on three methods (button push, PIN or NFC), none of which scale easily in an enterprise environment.

3.8. Define and Advertise a BRSKI-specific AKM in RSNE

[IEEE802.11i] introduced the RSNE element which allows an SSID to advertise multiple authentication mechanisms. A new Authentication and Key Management (AKM) Suite could be defined that indicates the STA can use BRSKI mechanisms to authenticate against the SSID. The authentication handshake could be an [IEEE802.1X] handshake, possibly leveraging an EAP-BRSKI mechanism, the key thing here is that a new AKM is defined and advertised to indicate the specific BRSKI-capable EAP method that is supported by [IEEE802.1X], as opposed to the current [IEEE802.1X] AKMs which give no indication of the supported EAP mechanisms. It is clear that such method would limit the SSID to BRSKI-supporting clients. This would require an additional SSID specifically for BRSKI clients. As such, this solution also suffers from the limitations mentioned about additional overhead. Additionally, this mechanism suffers from the limitations outlined in Section 2.1 - it does nothing to prevent a device attempting to enroll against an incorrect network.
3.9. Wi-Fi Device Provisioning Profile

The [DPP] specification, also known as Wi-Fi Easy Connect, defines how an entity that is already trusted by a network can assist an untrusted entity in enrolling with the network. The description below assumes the [IEEE802.11] network is in infrastructure mode. DPP introduces multiple key roles including:

- **Configurator**: A logical entity that is already trusted by the network that has capabilities to enroll and provision devices called Enrollees. A Configurator may be a STA or an AP.

- **Enrollee**: A logical entity that is being provisioned by a Configurator. An Enrollee may be a STA or an AP.

- **Initiator**: A logical entity that initiates the DPP Authentication Protocol. The Initiator may be the Configurator or the Enrollee.

- **Responder**: A logical entity that responds to the Initiator of the DPP Authentication Protocol. The Responder may be the Configurator or the Enrollee.

In the DPP model, a common Configurator and Initiator is an app running on a trusted smartphone. This process is manual, and each device is treated individually. In order to support a plug and play model for installation of a large number of devices, where each device is simply powered up for the first time and automatically discovers the Wi-Fi network without the need for a helper or supervising application, then this implies that the AP must perform the role of the Configurator and the device or STA performs the role of Enrollee. Note that the AP may simply proxy DPP messages through to a backend WLC, but from the perspective of the device, the AP is the Configurator.

The DPP specification also mandates that the Initiator must be bootstrapped the bootstrapping public key of the Responder. For BRSKI purposes, the DPP bootstrapping public key will be the [IEEE802.1AR] IDevID of the device. As the bootstrapping device cannot know in advance the bootstrapping public key of a specific operators network, this implies that the Configurator must take on the role of the Initiator. Therefore, the AP must take on the roles of both the Configurator and the Initiator.

At boot time, the device does not know which AP or which SSID is likely to provide DPP services. In the DPP model, the Configurator advertizes a special Authentication and Key Management (AKM) mode, DPP. Announcing this mode outside of onboarding windows might result in regular, non-DPP clients to fail to associate to a network which
AKM they do not recognize. As such, it is preferable that the DPP process be started after the device establishes a link with the access point. Therefore, DPP is likely not the best process to identify a supporting access point. Additionally, this mechanism suffers from the limitations outlined in Section 2.1 - it does nothing to prevent a device attempting to enroll against an incorrect network.

4. Potential Mutual Validation Options

When the bootstrapping device determines that one or more APs or SSIDs are available that provide support for BRSKI, with one or more of the mechanisms listed in section 3, then the device needs to determine which is the correct SSID. At the same time, an AP receiving signals from a bootstrapping device may need to verify if the need to determine if the device is attempting to connect the the correct network. In essence, this joint requirement means that BRSKi could be started immediately after the discovery phase. A case of mistaken identity (device attempting to join the wrong network) can be resolved with a round robin process, where the device fails the BRSKI process on the attempted network, then attempts BRSKi against the next candidate network. However, this process may result in wasted airtime and possible security exposure where an operator attempts to capture information about neighboring bootstrapping devices.

4.1. MAC Address Validation method

An alternative to the round robin mode is a primary selection mode where the device and the AP exchange mutual signs of knowledge about each other. This could be achieved using the standard 802.11 process, where the device would send a probe request using its real MAC address. This MAC address could be known to a central database and validated by the wireless infrastructure. This method has the merit of being simple. However, it is more and more common for devices with simple network stacks to use locally administered (and temporal) MAC addresses. This method only validates the device (not the infrastructure).

4.2. Vendor Token Validation method

An alternative to the MAC address method is to use a token, placed in an extension Information Element of the device probe request frame. This token would identify the device vendor. A limitation of this method is that, in some cases, neighboring networks may bootstrap devices from the same vendor. This method validates the vendor, but not the device. It also does not validate the infrastructure. It can be used as a coarse initial filtering mechanism.
4.3. Device Token Validation method

An alternative to the vendor token is to use a unique identifier for the device. However, as the transaction is exposed to eavesdropping, this method exposes the token. As such, the token should not be an element that can be compromised. The token can be the MAC address, if the device uses locally administered addresses for its probe requests. This method only validates the device (not the infrastructure).

4.4. Infrastructure Response Filtering

When additional filtering is required, the infrastructure may validate the additional information provided by the device, and either respond, if the additional information is computed to match the infrastructure knowledge, or ignore the request (no probe response) if the additional information does not match the infrastructure knowledge.

In some cases, the AP may not be able to access the database locally, and may need to forward the request (including the additional information provided by the device) to another system. In this case, the AP may respond with a frame that includes a GAS comeback value. This value indicates a delay after which the device should ask the question again. In that interval, the AP will query the infrastructure to obtain the additional information required. After expiration of the comeback interval, the device may send the probe request again, and the AP may respond or ignore the request, or request more time. It is understood that the device would accept a limited number of comeback requests (for example 3) and a limited comeback interval (for example no more than 3 seconds).

4.5. Infrastructure Validation Method

It is expected, when the device adds information to its probe request, that the infrastructure should only respond to those devices that have been validated by the infrastructure system. However, some systems may not be able to respond in time and may be configured to accept all requests. Additionally, bad actors may decide to accept any request. There may therefore be a need to mandate the infrastructure to return information that indicates proof of knowledge of the device. The following modes are envisioned:

- When the device uses its MAC address, or expresses its MAC address in an information element contained in the probe request, the infrastructure may be able to express its knowledge of the device serial number, and mention this serial number in the probe request.
response. As it may be needed to protect the serial number at this stage, the serial number could be encoded in a bloom filter.

- When the device uses a vendor token, the AP can only reply with another token identifying the same vendor, as the device itself is not known.

5. Potential Authentication Options

When the bootstrapping device determines which SSID to connect to, there are multiple potential options available for how the device authenticates with the network while bootstrapping. Several options are outlined in this section. This list is not exhaustive.

At a high level, authentication can generally be split into two phases using two different credentials:

- Pre-BRSKI: The device can use its [IEEE802.1AR] IDevID to connect to the network while executing the BRSKI flow
- Post-BRSKI: The device can use its [IEEE802.1AR] LDevID to connect to the network after completing BRSKI enrollment

The authentication options outlined in this document include:

- Unauthenticated Pre-BRSKI and EAP-TLS Post-BRSKI
- DPP Pre-BRSKI and EAP-TLS Post-BRSKI
- PSK or SAE Pre-BRSKI and EAP-TLS Post-BRSKI
- MAC Address Bypass Pre-BRSKI and EAP-TLS Post-BRSKI
- EAP-TLS Pre-BRSKI and EAP-TLS Post-BRSKI
- New DPP BRSKI mechanism
- New TEAP BRSKI mechanism
- New [IEEE802.1X] EAPOL-Announcements to encapsulate BRSKI prior to EAP-TLS Post-BRSKI

These mechanisms are described in more detail in the following sections. Note that any mechanisms leveraging [IEEE802.1X] are
MAC layer authentication mechanisms and therefore the SSID must advertise WPA2 capability.

When evaluating the multiple authentication options outlined below, care and consideration must be given to the complexity of the software state machine required in both devices and services for implementation.

5.1. Unauthenticated and Unencrypted or OWE Pre-BRSKI and EAP-TLS Post-BRSKI

The device connects to an unauthenticated network pre-BRSKI. The device connects to a network enforcing EAP-TLS post-BRSKI. The device uses its LDevID as the post-BRSKI EAP-TLS credential.

In the pre-BRSKI phase, the device may establish a secure connection with the AP using WPA3 to protect the BRSKI exchange from eavesdroppers. The pre-BRSKI phase can be protected, but is not authenticated.

5.2. DPP Pre-BRSKI and EAP-TLS post-BRSKI

The device can be provisioned with DPP for the pre-BRSKI phase, receiving the SSID value and optionally a temporal PSK. It should be noted that the device at that point is not untampered anymore. However, the configuration is temporal and limited. In a WPA3 network, when DPP from a mobile (e.g. smartphone) is used, the DPP process may provision the SSID and leave the device to use OWE for its connection to the AP.

Alternatively, when DPP is processed through the AP in an automated fashion, the AP first establishes an OWE connection with the device. Through this encrypted connection, the AP provides the SSID and the temporal PSK value.

5.3. PSK or SAE Pre-BRSKI and EAP-TLS Post-BRSKI

The device connects to a network enforcing PSK pre-BRSKI. If DPP is not used, the PSK may be factory-set (default PSK) or provisioned by direct action on the device. Neither of these modes is preferred as factory-defaults are weak and direct interaction with the device does not allow for massive automated bootstrapping. After the PSK-based pre-BRSKI connection, the device connects to a network enforcing EAP-TLS post-BRSKI. The device uses the LDevID obtained via BRSKI as the post-BRSKI EAP-TLS credential.

When the device connects to the post-BRSKI network that is enforcing EAP-TLS, the device uses its LDevID as its credential. The device
should verify the certificate presented by the server during that 
EAP-TLS exchange against the trusted CA list it obtained during 
BRSKI.

If the [IEEE802.1X] network enforces a tunneled EAP method, for 
example [RFC7170], where the device must present an additional 
credential such as a password, the mechanism by which that additional 
credential is provisioned on the device for post-BRSKI authentication 
is out-of-scope of this version of this document. NAI Realm may be 
used to advertise the EAP methods being enforced by an SSID. It is 
to be determined if guidelines should be provided on use of NAI Realm 
for advertising EAP method in order to streamline BRSKI.

5.4. MAC Address Bypass Pre-BRSKI and EAP-TLS Post-BRSKI

Many AAA server state machine logic allows for the network to 
fallback to MAC Address Bypass (MAB) when initial authentication 
against the network fails. If the device does not present a valid 
credential to the network, then the network will check if the 
device’s MAC address is whitelisted. If it is, then the network may 
grant the device access to a network segment that will allow it to 
complete the BRSKI flow and get provisioned with an LDevID. Once the 
device has an LDevID, it can then reauthenticate against the network 
using its EAP-TLS and its LDevID.

5.5. EAP-TLS Pre-BRSKI and EAP-TLS Post-BRSKI

The device connects to a network enforcing EAP-TLS pre-BRSKI. The 
device uses its IDevID as the pre-BRSKI EAP-TLS credential. The 
device connects to a network enforcing EAP-TLS post-BRSKI. The 
device uses its LDevID as the post-BRSKI EAP-TLS credential.

When the device connects to a pre-BRSKI network that is enforcing 
EAP-TLS, the device uses its IDevID as its credential. The device 
should not attempt to verify the certificate presented by the server 
during that EAP-TLS exchange, as it has not yet discovered the local 
domain trusted CA list.

When the device connects to the post-BRSKI network that is enforcing 
EAP-TLS, the device uses its LDevID as its credential. The device 
should verify the certificate presented by the server during that 
EAP-TLS exchange against the trusted CA list it obtained during 
BRSKI.

Again, if the post-BRSKI network enforces a tunneled EAP method, the 
mechanism by which that second credential is provisioned on the 
device is out-of-scope of this version of this document.
5.6. New DPP BRSKI mechanism

BRSKI can be integrated into the DPP choreography, in three modes:

- When a local commissioning tool is used (e.g. application on a mobile device), the standard DPP process is used for the configurator to establish a trusted connection to the enrolee (the bootstrapping device), over Bluetooth, NFC, Wi-Fi or other means defined by DPP. The configurator then provision the bootstrapping device with the target SSID, but also installs on the device the TrustAnchor. The bootstrapping device then connects to the target SSID using EAP-BRSKI (EST). The query is relayed to the registrar, which validates the device identity. An EAP-Success message is then returned to the access point.

- When the commissioning tool is not mobile and not interacting directly with the bootstrapping device, identifiers for the device may be fed into an authentication database (e.g. serial number, MAC address, DPP key, device-specific factory-set PSK or other). Upon device request (probe request with request for network proof of knowledge), the AP retrieves one or more of these parameters from the authentication database, and uses them to provide proof of knowledge to the device. Once trust is established, a temporal trusted link is established between the device and the AP (using DPP parameters or OWE) and the AP provisions the device with the SSID. The device then connects to the target SSID using EAP-BRSKI as above.

- When the authentication server has reachability to the MASA server, the process above is started. As the device conenctst to the target SSID, its identity is not only validated by the authentication server, but the authentication server also initiates a voucher request to the MASA server. The exchange between the bootstrapping device and the authentication server, now in possession of the voucher, continues as per [I-D.ietf-anima-bootstrapping-keyinfra].

5.7. New TEAP BRSKI mechanism

New TEAP TLVs are defined to transport BRSKI messages inside an outer EAP TLS tunnel such as TEAP [RFC7170]. [I-D.lear-eap-teap-brski] outlines a proposal for how BRSKI messages could be transported inside TEAP TLVs. At a high level, this enables the device to obtain an LDevID during the Layer 2 authentication stage. This has multiple advantages including:

- avoids the need for the device to potentially connect to two different SSIDs during bootstrap
o the device only needs to handle one authentication mechanism
during bootstrap

o the device only needs to obtain one IP address, which it obtains
after BRSKI is complete

o avoids the need for the device to have to disconnect from the
network, reset its network stack, and reconnect to the network

o potentially simplifies network policy configuration

There are two suboptions to choose from when tunneling BRSKI messages
inside TEAP:

o define new TLVs for transporting BRSKI messages inside the TEAP
tunnel

o define a new EAP BRSKI method type that is tunneled within the
outer TEAP method

This section assumes that new TLVs are defined for transporting BRSKI
messages inside the TEAP tunnel and that a new EAP BRSKI method type
is not defined.

The device discovers and connects to a network enforcing TEAP. A
high level TEAP with BRSKI extensions flow would look something like:

o Device starts the EAP flow by sending the EAP TLS ClientHello
message

o EAP server replies and includes CertificateRequest message, and
may specify certificate_authorities in the message

o if the device has an LDevID and the LDevID issuing CA is allowed
by the certificate_authorities list (i.e. the issuing CA is
explicitly included in the list, or else the list is empty) then
the device uses its LDevID to establish the TLS tunnel

o if the device does not have an LDevID, or certificate_authorities
prevents it using its LDevID, then the device uses its IDevID to
establish the TLS tunnel

o if certificate_authorities prevents the device from using its
IDevID (and its LDevID if it has one) then the device fails to
connect

The EAP server continues with TLS tunnel establishment:
If the device certificate is invalid or expired, then the EAP server fails the connection request.

If the device certificate is valid but is not allowed due to a configured policy on the EAP server, then the EAP server fails the connection request.

If the device certificate is accepted, then the EAP server establishes the TLS tunnel and starts the tunneled EAP-BRSKI procedures.

At this stage, the EAP server has some policy decisions to make:

- If network policy indicates that the device certificate is sufficient to grant network access, whether it is an LDevID or an IDevID, then the EAP server simply initiates the Crypto-Binding TLV and ‘Success’ Result TLV exchange. The device can now obtain an IP address and connect to the network.

- The EAP server may instruct the device to initialise a full BRSKI flow. Typically, the EAP server will instruct the device to initialize a BRSKI flow when it presents an IDevID, however, the EAP server may instruct the device to initialize a BRSKI flow even if it presented a valid LDevID. The device sends all BRSKI messages, for example ‘requestvoucher’, inside the TLS tunnel using new TEAP TLVs. Assuming the BRSKI flow completes successfully and the device is issued an LDevID, the EAP server completes the exchange by initiating the Crypto-Binding TLV and ‘Success’ Result TLV exchange.

Once the EAP flow has successfully completed, then:

- Network policy will automatically assign the device to the correct network segment.

- The device obtains an IP address.

- The device can access production service.

It is assumed that the device will automatically handle LDevID certificate reenrolment via standard EST [RFC7030] outside the context of the EAP tunnel.

An item to be considered here is what information is included in Beacon or Probe Response frames to explicitly indicate that [IEEE802.1X] authentication using TEAP supporting BRSKI extensions is allowed. Currently, the RSNE included in Beacon and Probe Response frames can only indicate [IEEE802.1X] support.
5.8. New IEEE 802.11 Authentication Algorithm for BRSKI and EAP-TLS Post-BRSKI

[IEEE802.11] supports multiple authentication algorithms in its Authentication frame including:

- Open System
- Shared Key
- Fast BSS Transition
- Simultaneous Authentication of Equals

Shared Key authentication is used to indicate that the legacy WEP authentication mechanism is to be used. Simultaneous Authentication of Equals is used to indicate that the Dragonfly-based shared passphrase authentication mechanism introduced in [IEEE802.11s] is to be used. One thing that these two methods have in common is that a series of handshake data exchanges occur between the device and the AP as elements inside Authentication frames, and these Authentication exchanges happen prior to [IEEE802.11] Association.

It would be possible to define a new Authentication Algorithm and define new elements to encapsulate BRSKI messages inside Authentication frames. For example, new elements could be defined to encapsulate BRSKI requestvoucher, voucher and voucher telemetry JSON messages. The full BRSKI flow completes and the device gets issued an LDevID prior to associating with an SSID, and prior to doing full [IEEE802.1X] authentication using its LDevID.

The high level flow would be something like:

- SSID Beacon / Probe Response indicates in RSNE that it supports BRSKI based Authentication Algorithm
- SSIDs could also advertise that they support both BRSKI based Authentication and [IEEE802.1X]
- device discovers SSID via suitable mechanism
- device completes BRSKI by sending new elements inside Authentication frames and obtains an LDevID
- device associates with the AP
- device completes [IEEE802.1X] authentication using its LDevID as credential for EAP-TLS or TEAP
5.9. New IEEE 802.1X EAPOL-Announcements to encapsulate BRSKI and EAP-TLS Post-BRSKI

[IEEE802.1X] defines multiple EAPOL packet types, including EAPOL-Announcement and EAPOL-Announcement-Req messages. EAPOL-Announcement and EAPOL-Announcement-Req messages can include multiple TLVs. EAPOL-Announcement messages can be sent prior to starting any EAP authentication flow. New TLVs could be defined to encapsulate BRSKI messages inside EAPOL-Announcement and EAPOL-Announcement-Req TLVs. For example, new TLVs could be defined to encapsulate BRSKI requestvoucher, voucher and voucher telemetry JSON messages. The full BRSKI flow could complete inside EAPOL-Announcement exchanges prior to sending EAPOL-Start or EAPOL-EAP messages.

The high level flow would be something like:

- SSID Beacon / Probe Response indicates somehow in RSNE that it supports [IEEE802.1X] including BRSKI extensions.
- device connects to SSID and completes standard Open System Authentication and Association
- device starts [IEEE802.1X] EAPOL flow and uses new EAPOL-Announcement frames to encapsulate and complete BRSKI flow to obtain an LDevID
- device completes [IEEE802.1X] authentication using its LDevID as credential for EAP-TLS or TEAP

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

The mechanisms described in this document rely on BRSKI. As such, the same security considerations are applicable to this document as they are in [I-D.ietf-anima-bootstrapping-keyinfra].

Additionally, the Wireless LAN presents a unique DOS attack vector, as endpoints contend for the shared medium on a completely egalitarian basis with the AP. This means that any wireless device could potentially monopolize the air by constantly sending frames. This would prevent the bootstrapping device, or the infrastructure, to complete their exchange and would make the BRSKI process fail. This risk is inherent to the nature of 802.11 transmissions, and can only be mitigated by physical access control to the cell area. Such attack is also easily detected.
Also, initial exchanges between the bootstrapping device and the AP are not protected. Whenever a unicast communication is initiated between a bootstrapping device and an AP in an attempt to start active bootstrapping or provisioning, the link should first be protected whenever possible, for example with OWE.

7.1. Client side exposure

The discovery mechanism imposes that the bootstrapping device and the infrastructure must exchange messages to be aware of each other's existence. If these messages are generic, then the bootstrapping device has no mechanism to distinguish the correct SSID from a neighboring SSID. The bootstrapping device then is faced with two options:

- Try all possible SSIDs in a round-robin fashion. By doing so, the bootstrapping device will potentially expose parameters to the wrong SSID and infrastructure. Although such exposure is unlikely to result in device compromission, it will still expose unnecessarily device parameters to the wrong network. As such, it is recommended that a pre-BRSKI filtering mechanism be implemented to avoid this exposure, conducting the bootstrapping device to only start the BRSKI process with an SSID that has been confirmed to be a likely correct candidate.

- When the bootstrapping device attempts to proceed to an SSID filtering, it may need to expose parameters to allow for the infrastructure to respond and provide a proof of knowledge. If this mechanism is implemented, the bootstrapping device should only expose information that is not sufficient to acquire complete knowledge of the bootstrapping device. For example, the bootstrapping device should not send both its serial number and MAC address, but should only expose an element that has low security value (such as a MAC address), and only in scenarios where the infrastructure has to respond with another element that will confirm to the bootstrapping device that it is communicating with the correct infrastructure.

7.2. Infrastructure side exposure

The general choreography of 802.11 networks imply that the infrastructure advertizes capabilities and support for specific features through beacons and probe responses. As such, the AP is likely to have to expose its support for BRSKI. This exposure is not a security concern.

When the infrastructure is requested to provide pre-BRSKI proof of knowledge, it has to process a frame received from an unknown
candidate device and either respond (if the device is found to be known), delay the response (if additional processing is needed) or ignore the request. Each of these behaviors may be tested by a rogue device in an attempt to gain information about the wireless infrastructure. It is therefore recommended that the proof of knowledge test should only focus on parameters specific to a particular device, and not to parameters generally applicable to multiple devices (for example parameters that would apply to multiple devices of one or more vendors).

8. Informative References


Appendix A.  IEEE 802.11 Primer

A.1. IEEE 802.11i

802.11i-2004 is an IEEE standard from 2004 that improves connection security. 802.11i-2004 is incorporated into 802.11-2014. 802.11i defines the Robust Security Network IE which includes information on:

- Pairwise Cipher Suites (WEP-40, WEP-104, CCMP-128, etc.)
- Authentication and Key Management Suites (PSK, 802.1X, etc.)
The RSN IEs are included in Beacon and Probe Response frames. STAs can use this frame to determine the authentication mechanisms offered by a particular AP e.g. PSK or 802.1X.

A.2. IEEE 802.11u

802.11u-2011 is an IEEE standard from 2011 that adds features that improve interworking with external networks. 802.11u-2011 is incorporated into 802.11-2016.

STAs and APs advertise support for 802.11u by setting the Interworking bit in the Extended Capabilities IE, and by including the Interworking IE in Beacon, Probe Request and Probe Response frames.

The Interworking IE includes information on:

- Access Network Type (Private, Free public, Chargeable public, etc.)
- Internet bit (yes/no)
- ASRA (Additional Step required for Access - e.g. Acceptance of terms and conditions, On-line enrollment, etc.)

802.11u introduced Access Network Query Protocol (ANQP) which enables STAs to query APs for information not present in Beacons/Probe Responses.

ANQP defines these key IEs for enabling the STA to determine which network to connect to:

- Roaming consortium IE: includes the Organization Identifier(s) of the roaming consortium(s). The OI is typically provisioned on cell phones by the SP, so the cell phone can automatically detect 802.11 networks that provide access to its SP’s consortium.

- 3GPP Cellular Network IE: includes the Mobile Country Code (MCC) and Mobile Network Code (MNC) of the SP the AP provides access to.

- Network Access Identifier Realm IE: includes [RFC4282] realm names that the AP provides access to (e.g. wifi.service-provider.com). The NAI Realm IE also includes info on the EAP type required to access that realm e.g. EAP-TLS.

- Domain name IE: the domain name(s) of the local AP operator. Its purpose is to enable a STA to connect to a domain operator that may have a roaming agreement with STA’s Service Provider.
STAs can use one or more of the above IEs to make a suitable decision on which SSID to pick.

HotSpot 2.0 is an example of a specification built on top of 802.11u and defines 10 additional ANQP elements using the standard vendor extensions mechanisms defined in 802.11. It also defines a HS2.0 Indication element that is included in Beacons and Probe Responses so that STAs can immediately tell if an SSID supports HS2.0.

Authors’ Addresses

Owen Friel
Cisco
Email: ofriel@cisco.com

Eliot Lear
Cisco
Email: lear@cisco.com

Jerome Henry
Cisco
Email: jerhenry@cisco.com

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

This document discusses the enhancement of automated bootstrapping of a remote secure key infrastructure (BRSKI) to operate in domains featuring no or only timely limited connectivity to backend services offering enrollment functionality like a Public Key Infrastructure (PKI). In the context of deploying new devices the design of BRSKI allows for online (synchronous object exchange) and offline interactions (asynchronous object exchange) with a manufacturer’s authorization service. It utilizes a self-contained voucher to transport the domain credentials as a signed object to establish an initial trust between the pledge and the deployment domain. The currently supported enrollment protocol for request and distribution of deployment domain specific device certificates provides only limited support for asynchronous PKI interactions. This memo motivates support of self-contained objects also for certificate management by using an abstract notation to allow off-site operation of PKI services, with only limited connectivity to the pledge deployment domain. This addresses specifically scenarios, in which the deployment domain of a pledge does not perform the final authorization of a certification request and rather delegates this decision to an operator backend. The goal is to enable the usage of existing and potentially new PKI protocols supporting self-containment for certificate management.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
2. Terminology .................................................. 5
3. Scope of solution ............................................. 6
   3.1. Supported environment ................................. 6
   3.2. Application Examples .................................. 6
      3.2.1. Rolling stock ..................................... 6
      3.2.2. Building automation ............................... 7
      3.2.3. Substation automation ............................. 7
      3.2.4. Electric vehicle charging infrastructure ......... 7
   3.3. Requirements for asynchronous operation ............... 8
4. Architectural Overview ........................................ 8
   4.1. Secure Imprinting using Vouchers ....................... 11
   4.2. Addressing ............................................. 11
5. Protocol Flow ................................................ 12
   5.1. Pledge - Registrar discovery and voucher exchange .... 12
   5.2. Registrar - MASA voucher exchange .................... 14
   5.3. Pledge - Registrar - RA/CA certificate enrollment ...... 14
6. IANA Considerations .......................................... 16
7. Privacy Considerations ....................................... 16
8. Security Considerations ...................................... 16
9. Acknowledgements ............................................. 16
10. References .................................................. 16
    10.1. Normative References ................................ 16
    10.2. Informative References .............................. 17
Authors’ Addresses ............................................. 17
1. Introduction

BRSKI as defined in [I-D.ietf-anima-bootstrapping-keyinfra] specifies a solution for secure zero-touch (automated) bootstrapping of devices (pledges) in a target deployment domain. This includes the discovery of network elements in the deployment domain, time synchronization, and the exchange of security information necessary to adopt a pledge as new network and application element. Security information about the deployment domain, specifically the deployment domain certificate (domain root certificate), is exchanged utilizing vouchers as defined in [RFC8366]. These vouchers are self-contained objects, which may be provided online (synchronous) or offline (asynchronous) via the domain registrar to the pledge and originate from a manufacturer’s authorization service (MASA). The manufacturer signed voucher contains the target domain certificate and can be verified by the pledge due to the possession of a manufacturer root certificate. It facilitates the enrollment of the pledge in the deployment domain and is used to establish trust.

For the enrollment of devices BRSKI relies on EST [RFC7030] to request and distribute deployment domain specific device certificates. EST in turn relies on a binding of the certification request to an underlying TLS connection between the EST client and the EST server. The EST server is likely collocated with a registration authority (RA) or local registration authority (LRA). The binding to TLS is used to protect the exchange of a certification request (for an LDevID certificate) and to provide data origin authentication to support the authorization decision for processing the certification request. The TLS connection is mutually authenticated and the client side authentication bases on the pledge’s manufacturer issued device certificate (IDevID certificate). This approach requires an on-site availability of a PKI component and/or a local asset or inventory management system performing the authorization decision to issue a domain specific certificate to the pledge. This is due to the fact that the EST server terminates the security association with the pledge and thus the binding between the certification request and the authentication of the pledge. Moreover, it may also require to setup a new security association between the EST and the issuing RA/CA. This type of enrollment utilizing an online connection to the PKI is considered as synchronous enrollment.

For certain use cases on-site support of a RA/CA component and/or an asset management is not available and rather provided in a timely limited fashion or completely offline. This may be due to higher security requirements for the certification authority. This also means that a PKI component, performing the authorization decision for a certification request from a pledge may not be available on-site at
enrollment time. Enrollment, which cannot be performed in a (timely) consistent fashion is considered as asynchronous enrollment in this document. In this case a support of a store and forward functionality of certification request together with the requester authentication information is necessary, to enable the processing of the request at a later point in time. A similar situation may occur through network segmentation, which is utilized in industrial systems to separate certain tasks. Here, a similar requirement arises if the communication channel carrying the requester authentication is terminated before the RA/CA. If a second communication channel is opened to forward the certification request to the issuing CA, the requester authentication information needs to be bound to the certification request. For both cases, it is assumed that the requester authentication information is utilized in the process of authorization of a certification request. There are different options to perform store and forward of certification requests:

- Providing a trusted component (e.g., an LRA) in the deployment domain, which handles the storage of the certification request combined with the requester authentication information (the IDevID) and potentially the information about a successful proof of possession in a way prohibiting changes to the combined information. Note that the assumption is that the information elements are not cryptographically bound together. Once the PKI functionality (RA/CA)) is available, the trusted component forwards the certification request together with the originator information and the information about the successful proof of possession as triple to the off-site PKI for further processing. It is assumed that the off-site PKI in this case relies on the local authentication result and thus on the authorization and issues the requested certificate. In BRSKI the trusted component may be the EST server residing co-located with the registrar in the deployment domain.

- Utilization of a self-contained object for the certification request, which cryptographically binds the requester authentication information to the certification request. This approach reduces the necessary trust in a domain component to storage and delivery. Unauthorized modifications can be detected during the verification of the cryptographic binding of the certification request in the off-site PKI.

This document targets environments, in which connectivity to the PKI functionality is only temporary or not directly available by specifying support for handling asynchronous objects supporting enrollment. As it is intended to enhance BRSKI it is named BRSKI-AE, where AE stands for asynchronous enrollment. Note that BRSKI-AE is also intended to be applicable for synchronous enrollment, e.g., if a
connection carrying the requester authentication is terminated before
the actual registration authority.

/* to be clarified: Describe as abstract type in Yang? */

The ultimate goal is to allow existing certificate management
protocols to be applied or to allow other types of encoding for the
certificate management information exchange.

Note that in contrast to BRSKI, BRSKI-AE assumes support of multiple
enrollment protocols on the infrastructure side, allowing the pledge
manufacturer to select the most appropriate.

As BRSKI, BRSKI-AE results in the pledge storing a X.509 root domain
certificate sufficient for verifying the domain registrar / proxy
identity. In the process a TLS connection is established that can be
directly used for certification request/response exchanges. The
certification request may be stored on the domain registrar / proxy
until connectivity to the PKI (issuing CA) becomes available. With
this, BRSKI-AE supports the automated mechanism for asynchronous
enrollment of a pledge in a deployment domain utilizing a voucher of
the pledge manufacturer resulting in a domain specific X.509 device
certificate (LDevID certificate) available on the pledge.

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].

This document relies on the terminology defined in
[I-D.ietf-anima-bootstrapping-keyinfra]. The following terms are
defined additionally:

CA: Certification authority, issues certificates.

RA: Registration authority, an optional system component to which a
CA delegates certificate management functions such as
authorization checks.

LRA: Local registration authority, an optional RA system component
with proximity to end entities.

IED: Intelligent Electronic Device (in essence a pledge).

on-site: Describes a component or service or functionality available
in the target deployment domain.
off-site: Describes a component or service or functionality available in an operator domain different from the target deployment domain. This may be a central side, to which only a temporarily connection is available or which is in a different administrative domain.

asynchronous communication: Describes a timely interrupted communication between an end entity and a PKI component.

synchronous communication: Describes a timely uninterrupted communication between an end entity and a PKI component.

3. Scope of solution

3.1. Supported environment

This solution is intended to be used in environments with no or only limited connectivity to backend services provided in the operator domain. Beyond others this comprises cases in which:

- there is no registration authority available in the deployment domain. The connectivity to the registration authority may only be temporarily available. A local store and forward device is used for the communication with the backend services.

- authoritative actions of a local registration authority are limited and may not comprise local authorization of certification requests of enrolling pledges. Final authorization is done at the registration authority residing in the operator domain.

- the target deployment domain already uses a certificate management approach that shall be kept consistent throughout the lifecycle.

3.2. Application Examples

The following examples are intended to motivate the support of different enrollment approaches in general and asynchronous enrollment specifically, by introducing industrial applications cases, which could leverage BRSKI as such but also require support of asynchronous operation as intended with BRSKI-AE.

3.2.1. Rolling stock

Rolling stock or railroad cars contain a variety of sensors, actuators, and controller, which communicate within the railroad car but also exchange information between railroad cars building a train or with a backend. These devices are typically unaware of backend connectivity. Managing certificates may be done during maintenance
cycles of the railroad car, but can already be prepared during operation. The preparation may comprise the generation of certificate signing requests, to apply for a new or an updated domain specific device certificate. The authorization of the certificate signing request is done using inventory information available in the backend.

/* to be done: more information to be provided */

3.2.2. Building automation

Detached building equipped with sensor, actuators, and controllers connected to centralized building management system. Limited/no connectivity to backend during the installation phase and even later. (Example: School, etc.)

/* to be done: more information to be provided */

3.2.3. Substation automation

In substation automation a control center typically hosts PKI services to issue certificates for IEDs in a substation. Communication between the substation and control center is typically done through a proxy/gateway/DMZ, which terminates protocol flows. Note that NERC CIP (reference to be included) requires inspection of protocols at the boundary of a security perimeter. In addition, security in substation automation assumes central support of different enrollment protocols to facilitate the capabilities of IEDs from different vendors. The IEC standard IEC62351-9 [IEC-62351-9] specifies the mandatory support of two enrollment protocols, SCEP [I-D.gutmann-scep] and EST [RFC7030] for the infrastructure side, while the IEDs must only support one of the two.

3.2.4. Electric vehicle charging infrastructure

For the electric vehicle charging infrastructure protocols have been defined for the interaction between the electric vehicle and the charging spot (e.g., ISO 15118 [ISO-IEC-15118-2]) as well as between the charging spot and the operator backend (e.g. OCPP [OCPP]). Depending on the charging model, unilateral or mutual authentication is required. In both cases the charging spot authenticates using an X.509 certificate. The management of this certificate depends (beyond others) on the selected backend connectivity protocol. In case of OCPP there is the desire to have a single communication protocol between the charging spot and the backend carrying all information to control and manage the charging operations and the charging spot itself. This means that the certificate management is intended to be handled in-band of OCPP. This requires to be able to
encapsulate the certificate management exchanges in a transport independent way. Self-containment will ease this by allowing the transport without a separate communication protocol.

3.3. Requirements for asynchronous operation

Based on the supported environment described in Section 3.1 and the motivated application examples described in Section 3.2 the following base requirements are derived:

- Certificate management exchanges (e.g., certification request and certification response message(s)) are ideally carried in a container protecting at least integrity of the exchanges and providing source authentication. /* to be clarified: reference to PKCS#10 or CRMF to be used? */

- The container with the certification request should provide a proof of possession of corresponding private key. Note: this is typically provided by the existing enrollment protocols and is stated here for completeness if a different approach (encoding, transport) is desired.

- The container with the certification request should support a cryptographic binding to an existing credential known to the operator domain. /* to be clarified: reference to existing enrollment protocols EST, CMC, CMP, SCEP to be used? */

- The container with the certification request should support direct protection using an existing credential on the pledge verifiable in the operator domain. /* to be clarified: reference to CMS or CMP to be used? */

4. Architectural Overview

The intended architecture for supporting asynchronous enrollment relies architecture defined in BRSKI [I-D.ietf-anima-bootstrapping-keyinfra] with certain changes as shown in the placement or enhancements of the logical elements in Figure 1.
Figure 1: Architecture overview of BRSKI-AE

The architecture overview in Figure 1 utilizes the same logical elements as BRSKI but with a different placement in the architecture for some of the elements in terms of connected domains. The main difference is the placement of the PKI RA/CA component as well as the connectivity of the RA/CA with an inventory management system. Both are placed in the operator domain, which may have no or only temporary connectivity to the deployment domain of the pledge. Based
on the assumed connectivity of the deployment domain, the MASA interaction may also be done asynchronous to the actual deployment domain. The following list describes the deployment domain components:

- Join Proxy: same functionality as described in BRSKI

- Domain Registrar / Proxy: In general the domain registrar / proxy has a similar functionality regarding the imprinting of the pledge in the deployment domain. Differences arise, if the deployment domain has temporary or no connectivity to an operator domain and/or the manufacturers MASA. There may be use cases, in which the (domain) registrar may even be operated in the operator domain. /* to do: needs more description */

  * Voucher exchange: The voucher exchange with the MASA is performed as described in BRSKI [I-D.ietf-anima-bootstrapping-keyinfra]. If the voucher exchange is facilitated by the operator domain, additional description is necessary. In Figure 1 this is characterized by indicating an alternative path for the voucher request/response interaction.

  * Certificate enrollment: For the pledge enrollment the domain registrar in the deployment domain is expected to support the authorization of the pledge to be part of the domain, but not necessarily to authorize the certification request provided during enrollment. This may be due to lack of authorization information in the deployment domain. If the authorization is done in the operator domain, the domain registrar is used as store and forward component (or proxy) of the certification requests. To enable this, the domain registrar needs functionality enhancements regarding the support of alternative enrollment approaches supporting self-containment. To support alternative enrollment approaches (protocoi, encodings), it is necessary to enhance the addressing scheme at the domain registrar. The communication channel between the pledge and the domain registrar may be similarly described within the same "/.well-known" tree and may result for instance in "/.well-known/enrollment-variant/request".

The following list describes the vendor related components/service outside the deployment domain:

- MASA: general functionality as described in BRSKI. Assumption that the interaction may be done synchronous and asynchronous based on the general assumption that the deployment domain has
limited outside connectivity. Note: additional steps for offline operation may need to be defined.

- Ownership tracker: as defined in BRSKI.

The following list describes the operator related components/service outside the deployment domain in the operator domain:

- (Domain) registrar: Optional component if the deployment domain does not feature a domain registrar but only a proxy. In this case it is involved in the certification request processing and is assumed to be co-located with the PKI RA. In addition, the registrar may be involved in the voucher exchange with the MASA. /* to be done: more elaboration necessary */

- PKI RA: Perform certificate management functions (validation of certification requests, interaction with inventory/asset management for authorization, etc.) for issuing, updating, and revoking certificates for a domain as a centralized infrastructure for the operator.

- PKI CA: Perform certificate generation by signing the certificate structure management.

- Inventory (asset) management: contains information about the known devices belonging to the operator. Specifically, the inventory is used to provide the information to authorize issuing a certificate based on the certification request of the pledge. Note: the communication between the inventory (asset) management and the PKI components (RA/CA) in the operator domain are out of scope for this document.

4.1. Secure Imprinting using Vouchers

/* to be done, should contain - review of the domain registrar - MASA interaction regarding offline operation - changes to the enrollment interaction through off-site RA/CA support */

4.2. Addressing

For the provisioning of different enrollment options at the domain registrar, the addressing approach of BRSKI using a "/.well-known" tree from [RFC5785] is enhanced.

/* to be done: Description of "/.well-known/enrollment-protocol/request" in which enrollment-protocol may be an already existing protocol like "est" or "scep" or "cmp" or a newly defined protocol. */
5. Protocol Flow

Based on BRSKI and the architectural changes the original protocol flow is divided into three phases showing commonalities and differences to the original approach as depicted in the following.

- Discovery phase (same as BRSKI)
- Voucher exchange with deployment domain registrar (may have changes due the handling of phases without communication to the operator domain).
- Enrollment phase (changed to accompany the asynchronous operation)

5.1. Pledge - Registrar discovery and voucher exchange

/* to be done: description of unchanged BRSKI approach */
Figure 2: Pledge discovery of domain registrar discovery and voucher exchange

/* to be done: - discuss call flow in the context of asynchronous operation, when the domain registrar works as proxy. The voucher waiting indication can be used in this way to inform the pledge not to expect an immediate response (may contain the time for the polling) - may utilize a parallel provisioning of a voucher request and a certification request by the pledge. - both may be provided
when the operator domain is available and processed sequentially by
the pledge, first the voucher, second the certification response */

5.2. Registrar - MASA voucher exchange

/* to be done: - clarification if BRSKI protocol sequence kept
unchanged - changes for complete offline operation may be necessary,
verify BRSKI document section 6.2. Pledge security reductions */

<table>
<thead>
<tr>
<th>Pledge</th>
<th>Circuit</th>
<th>Domain</th>
<th>Vendor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Join</td>
<td>Registrar</td>
<td>Service</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
<td>(JRC)</td>
<td>(MASA)</td>
</tr>
</tbody>
</table>

Figure 3: Domain registrar - MASA voucher exchange

5.3. Pledge - Registrar - RA/CA certificate enrollment

/* to be done: overview description of operation */
Figure 4: Certificate enrollment

- Cert Request: certification request message (to be done: reference to PKCS#10 or CRMF, proof of possession, pledge authentication)

- Cert Response: certification response message containing the requested certificate and potentially further information like certificates of intermediary CAs on the certification path.

- Cert Waiting: waiting indication for the pledge to retry after a given time.

- Cert Polling: querying the registrar, if the certificate request was already processed; can be answered either with another Cert Waiting, or a Cert Response.

- Cert Confirm: confirmation message from pledge after receiving and verifying the certificate.

- PKI/Registrar Confirm: confirmation message from PKI/registrar about reception of the pledge’s certificate confirmation.
/* to be done: - investigation into handling of certificate request retries - message exchange description - confirmation message (necessary? optional? from Registrar and/or PKI?) */

6. IANA Considerations

This document requires the following IANA actions:

/* to be done: clarification necessary */

7. Privacy Considerations

/* to be done: clarification necessary */

8. Security Considerations

/* to be done: clarification necessary */

9. Acknowledgements

We would like to thank the various reviewers for their input, in particular ...

10. References

10.1. Normative References

[I-D.ietf-anima-bootstrapping-keyinfra]


10.2. Informative References

[I-D.gutmann-scep]
draft-gutmann-scep-13 (work in progress), January 2019.

[IEC-62351-9]
International Electrotechnical Commission, "IEC 62351 -
Power systems management and associated information
exchange - Data and communications security - Part 9:
Cyber security key management for power system equipment",

[ISO-IEC-15118-2]
International Standardization Organization / International
Electrotechnical Commission, "ISO/IEC 15118-2 Road
vehicles - Vehicle-to-Grid Communication Interface - Part
2: Network and application protocol requirements", ISO/
IEC 15118, April 2014.

[OCPP]
Open Charge Alliance, "Open Charge Point Protocol 2.0",
April 2018.

[RFC5785]
Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known
Uniform Resource Identifiers (URIs)", RFC 5785,
DOI 10.17487/RFC5785, April 2010,

Authors’ Addresses

Steffen Fries
Siemens AG
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

Email: steffen.fries@siemens.com
URI: http://www.siemens.com/

Hendrik Brockhaus
Siemens AG
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

Email: hendrik.brockhaus@siemens.com
URI: http://www.siemens.com/
An Autonomic Control Plane (ACP)
draft-ietf-anima-autonomic-control-plane-19

Abstract

Autonomic functions need a control plane to communicate, which depends on some addressing and routing. This Autonomic Management and Control Plane should ideally be self-managing, and as independent as possible of configuration. This document defines such a plane and calls it the "Autonomic Control Plane", with the primary use as a control plane for autonomic functions. It also serves as a "virtual out-of-band channel" for Operations Administration and Management (OAM) communications over a network that provides automatically configured hop-by-hop authenticated and encrypted communications via automatically configured IPv6 even when the network is not configured, or misconfigured.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction (Informative) ................................. 5
   1.1. Applicability and Scope .......................... 8
2. Acronyms and Terminology (Informative) .................. 9
3. Use Cases for an Autonomic Control Plane (Informative) ... 15
   3.1. An Infrastructure for Autonomic Functions ....... 15
   3.2. Secure Bootstrap over a not configured Network .... 16
   3.3. Data-Plane Independent Permanent Reachability .... 16
4. Requirements (Informative) .................................. 17
5. Overview (Informative) ...................................... 18
6. Self-Creation of an Autonomic Control Plane (ACP) (Normative) 20
   6.1. ACP Domain, Certificate and Network .............. 20
       6.1.1. Certificate ACP Domain Information Field .... 21
       6.1.2. ACP domain membership check ................. 24
       6.1.3. Trust Points and Trust Anchors .............. 26
       6.1.4. Certificate and Trust Point Maintenance .... 27
       6.1.4.1. GRASP objective for EST server ........... 27
       6.1.4.2. Renewal .................................... 29
       6.1.4.3. Certificate Revocation Lists (CRLs) ....... 29
       6.1.4.4. Lifetimes .................................. 30
       6.1.4.5. Re-enrollment ................................ 30
       6.1.4.6. Failing Certificates ....................... 31
   6.2. ACP Adjacency Table .................................... 32
   6.3. Neighbor Discovery with DULL GRASP .................. 33
   6.4. Candidate ACP Neighbor Selection ..................... 36
   6.5. Channel Selection ..................................... 36
   6.6. Candidate ACP Neighbor verification ................. 39
   6.7. Security Association protocols ....................... 39
       6.7.1. ACP via IKEv2 .................................. 39
           6.7.1.1. Native IPsec ............................ 39
           6.7.1.2. IPsec with GRE encapsulation .......... 40
       6.7.2. ACP via DTLS ................................... 40
       6.7.3. ACP Secure Channel Requirements ............... 41
   6.8. GRASP in the ACP ....................................... 42
       6.8.1. GRASP as a core service of the ACP ............ 42
       6.8.2. ACP as the Security and Transport substrate for GRASP 42
           6.8.2.1. Discussion ................................ 45
6.9. Context Separation .................................. 46
6.10. Addressing inside the ACP ............................. 47
  6.10.1. Fundamental Concepts of Autonomic Addressing .... 47
  6.10.2. The ACP Addressing Base Scheme ................ 48
  6.10.3. ACP Zone Addressing Sub-Scheme .................. 50
    6.10.3.1. Usage of the Zone-ID Field .................. 51
  6.10.4. ACP Manual Addressing Sub-Scheme ................. 52
  6.10.5. ACP Vlong Addressing Sub-Scheme ................ 53
  6.10.6. Other ACP Addressing Sub-Schemes ................. 54
  6.10.7. ACP Registrars .................................. 55
    6.10.7.1. Use of BRSKI or other Mechanism/Protocols ... 55
    6.10.7.2. Unique Address/Prefix allocation ............. 56
    6.10.7.3. Addressing Sub-Scheme Policies ............... 56
    6.10.7.4. Address/Prefix Persistence ................... 57
  6.10.7.5. Further Details ................................ 58
  6.11. Routing in the ACP ................................ 58
    6.11.1. RPL Profile ................................... 58
      6.11.1.1. Overview ................................... 58
      6.11.1.2. RPL Instances ............................... 60
      6.11.1.3. Storing vs. Non-Storing Mode ................. 60
      6.11.1.4. DAO Policy ................................ 60
      6.11.1.5. Path Metric ................................. 60
      6.11.1.6. Objective Function .......................... 61
      6.11.1.7. DODAG Repair ............................... 61
      6.11.1.8. Multicast ................................. 61
      6.11.1.9. Security ................................. 61
      6.11.1.10. P2P communications ........................ 61
      6.11.1.11. IPv6 address configuration ................ 61
      6.11.1.12. Administrative parameters .................. 62
      6.11.1.13. RPL Data-Plane artifacts ................... 62
      6.11.1.14. Unknown Destinations ....................... 62
  6.12. General ACP Considerations .......................... 62
    6.12.1. Performance ................................... 63
    6.12.2. Addressing of Secure Channels ................... 63
    6.12.3. MTU ......................................... 63
    6.12.4. Multiple links between nodes ................... 64
    6.12.5. ACP interfaces ................................ 64
  7. ACP support on L2 switches/ports (Normative) .......... 67
    7.1. Why (Benefits of ACP on L2 switches) ............... 67
    7.2. How (per L2 port DULL GRASP) ....................... 68
  8. Support for Non-ACP Components (Normative) ............ 70
    8.1. ACP Connect ...................................... 70
      8.1.1. Non-ACP Controller / NMS system ................ 70
      8.1.2. Software Components ............................ 72
      8.1.3. Auto Configuration ............................. 73
      8.1.4. Combined ACP/Data-Plane Interface (VRF Select) . 74
      8.1.5. Use of GRASP .................................. 75
    8.2. ACP through Non-ACP L3 Clouds (Remote ACP neighbors) 76
1. Introduction (Informative)

Autonomic Networking is a concept of self-management: Autonomic functions self-configure, and negotiate parameters and settings across the network. [RFC7575] defines the fundamental ideas and design goals of Autonomic Networking. A gap analysis of Autonomic
Networking is given in [RFC7576]. The reference architecture for
Autonomic Networking in the IETF is specified in the document
[I-D.ietf-anima-reference-model].

Autonomic functions need an autonomically built communications
infrastructure. This infrastructure needs to be secure, resilient
and re-usable by all autonomic functions. Section 5 of [RFC7575]
introduces that infrastructure and calls it the Autonomic Control
Plane (ACP). More descriptively it would be the "Autonomic
communications infrastructure for Management and Control". For
naming consistency with that prior document, this document continues
to use the name ACP though.

Today, the management and control plane of networks typically uses a
routing and forwarding table which is dependent on correct
configuration and routing. Misconfigurations or routing problems can
disrupt management and control channels. Traditionally, an out-of-
band network has been used to avoid or allow recovery from such
problems, or personnel are sent on site to access devices through
out-of-band management ports (also called craft ports, serial
console, management ethernet port). However, both options are
expensive.

In increasingly automated networks either centralized management
systems or distributed autonomic service agents in the network
require a control plane which is independent of the configuration of
the network they manage, to avoid impacting their own operations
through the configuration actions they take.

This document describes a modular design for a self-forming, self-
managing and self-protecting Autonomic Control Plane (ACP), which is
a virtual in-band network designed to be as independent as possible
of configuration, addressing and routing problems. The details how
this is achieved are described in Section 6. The ACP is designed to
remain operational even in the presence of configuration errors,
addressing or routing issues, or where policy could inadvertently
affect connectivity of both data packets or control packets.

This document uses the term "Data-Plane" to refer to anything in the
network nodes that is not the ACP, and therefore considered to be
dependent on (mis-)configuration. This Data-Plane includes both the
traditional forwarding-plane, as well as any pre-existing control-
plane, such as routing protocols that establish routing tables for
the forwarding plane.

The Autonomic Control Plane serves several purposes at the same time:
1. Autonomic functions communicate over the ACP. The ACP therefore directly supports Autonomic Networking functions, as described in [I-D.ietf-anima-reference-model]. For example, Generic Autonomic Signaling Protocol (GRASP - [I-D.ietf-anima-grasp]) runs securely inside the ACP and depends on the ACP as its "security and transport substrate".

2. A controller or network management system can use it to securely bootstrap network devices in remote locations, even if the (Data-Plane) network in between is not yet configured; no Data-Plane dependent bootstrap configuration is required. An example of such a secure bootstrap process is described in [I-D.ietf-anima-bootstrapping-keyinfra].

3. An operator can use it to log into remote devices, even if the network is misconfigured or not configured.

This document describes these purposes as use cases for the ACP in Section 3, it defines the requirements in Section 4. Section 5 gives an overview how the ACP is constructed.

The normative part of this document starts with Section 6, where the ACP is specified. Section 7 defines normative how to support ACP on L2 switches. Section 8 explains normative how non-ACP nodes and networks can be integrated.

The remaining sections are non-normative: Section 9 reviews benefits of the ACP (after all the details have been defined), Section 10 provides operational recommendations, Appendix A provides additional explanations and describes additional details or future standard or propriety extensions that were considered not to be appropriate for standardization in this document but were considered important to document. There are no dependencies against Appendix A to build a complete working and interoperable ACP according to this document.

The ACP provides secure IPv6 connectivity, therefore it can be used not only as the secure connectivity for self-management as required for the ACP in [RFC7575], but it can also be used as the secure connectivity for traditional (centralized) management. The ACP can be implemented and operated without any other components of autonomic networks, except for the GRASP protocol. ACP relies on per-link DULL GRASP (see Section 6.3) to autodiscover ACP neighbors, and includes the ACP GRASP instance to provide service discovery for clients of the ACP (see Section 6.8) including for its own maintenance of ACP certificates.

The document "Using Autonomic Control Plane for Stable Connectivity of Network OAM" [RFC8368] describes how the ACP alone can be used to...
provide secure and stable connectivity for autonomic and non-autonomic Operations Administration and Management (OAM) applications. That document also explains how existing management solutions can leverage the ACP in parallel with traditional management models, when to use the ACP and how to integrate with potentially IPv4 only OAM backends.

Combining ACP with Bootstrapping Remote Secure Key Infrastructures (BRSKI), see [I-D.ietf-anima-bootstrapping-keyinfra]) results in the "Autonomic Network Infrastructure" as defined in [I-D.ietf-anima-reference-model], which provides autonomic connectivity (from ACP) with fully secure zero-touch (automated) bootstrap from BRSKI. The ANI itself does not constitute an Autonomic Network, but it allows the building of more or less autonomic networks on top of it - using either centralized, Software Defined Networking- (SDN-)style (see [RFC7426]) automation or distributed automation via Autonomic Service Agents (ASA) / Autonomic Functions (AF) - or a mixture of both. See [I-D.ietf-anima-reference-model] for more information.

1.1. Applicability and Scope

Please see the following Terminology section (Section 2) for explanations of terms used in this section.

The design of the ACP as defined in this document is considered to be applicable to all types of "professionally managed" networks: Service Provider, Local Area Network (LAN), Metropolitan networks, Wide Area Network (WAN), Enterprise Information Technology (IT) and Operational Technology (OT) networks. The ACP can operate equally on layer 3 equipment and on layer 2 equipment such as bridges (see Section 7). The hop-by-hop authentication and confidentiality mechanism used by the ACP is defined to be negotiable, therefore it can be extended to environments with different protocol preferences. The minimum implementation requirements in this document attempt to achieve maximum interoperability by requiring support for multiple options depending on the type of device: IPsec, see [RFC4301], and datagram Transport Layer Security version 1.2 (DTLS), see [RFC6347]).

The implementation footprint of the ACP consists of Public Key Infrastructure (PKI) code for the ACP certificate, the GRASP protocol, UDP, TCP and TLS (for security and reliability of GRASP), the ACP secure channel protocol used (such as IPsec or DTLS), and an instance of IPv6 packet forwarding and routing via the Routing Protocol for Low-power and Lossy Networks (RPL), see [RFC6550], that is separate from routing and forwarding for the Data-Plane (user traffic).
The ACP uses only IPv6 to avoid complexity of dual-stack ACP operations (IPv6/IPv4). Nevertheless, it can without any changes be integrated into even otherwise IPv4-only network devices. The Data-Plane itself would not need to change, it could continue to be IPv4 only. For such IPv4 only devices, the IPv6 protocol itself would be additional implementation footprint only used for the ACP.

The protocol choices of the ACP are primarily based on wide use and support in networks and devices, well understood security properties and required scalability. The ACP design is an attempt to produce the lowest risk combination of existing technologies and protocols to build a widely applicable operational network management solution:

RPL was chosen because it requires a smaller routing table footprint in large networks compared to other routing protocols with an autonomically configured single area. The deployment experience of large scale Internet of Things (IoT) networks serves as the basis for wide deployment experience with RPL. The profile chosen for RPL in the ACP does not leverage any RPL specific forwarding plane features (IPv6 extension headers), making its implementation a pure control plane software requirement.

GRASP is the only completely novel protocol used in the ACP, and this choice was necessary because there is no existing suitable protocol to provide the necessary functions to the ACP, so GRASP was developed to fill that gap.

The ACP design can be applicable to (cpu, memory) constrained devices and (bitrate, reliability) constrained networks, but this document does not attempt to define the most constrained type of devices or networks to which the ACP is applicable. RPL and DTLS for ACP secure channels are two protocol choices already making ACP more applicable to constrained environments. Support for constrained devices in this specification is opportunistic, but not complete, because the reliable transport for GRASP (see Section 6.8.2) only specifies TCP/TLS). See Appendix A.9 for discussions about how future standards or proprietary extensions/variations of the ACP could better meet different expectations from those on which the current design is based including supporting constrained devices better.

2. Acronyms and Terminology (Informative)

[RFC Editor: WG/IETF/IESG review of the terms below asked for references between these terms when they refer to each other. The only option I could find RFC/XML to point to a hanging text acronym definition that also displays the actual term is the format="title" version, which leads to references such as ‘--"ACP domain certificate" ()’. I found no reasonable way to eliminate the
trailing ')' generated by this type of cross references. Can you please take care of removing these artefacts during editing (after conversion to nroff?). I also created a ticket to ask for an xml2rfc enhancement to avoid this in the future: https://trac.tools.ietf.org/tools/xml2rfc/trac/ticket/347.

[RFC Editor: Question: Is it possible to change the first occurrences of [RFCxxxx] references to "rfcxxxx title" [RFCxxxx]? the XML2RFC format does not seem to offer such a format, but I did not want to duplicate 50 first references - one reference for title mentioning and one for RFC number.]

In the rest of the document we will refer to systems using the ACP as "nodes". Typically such a node is a physical (network equipment) device, but it can equally be some virtualized system. Therefore, we do not refer to them as devices unless the context specifically calls for a physical system.

This document introduces or uses the following terms (sorted alphabetically). Terms introduced are explained on first use, so this list is for reference only.

ACP: "Autonomic Control Plane". The Autonomic Function as defined in this document. It provides secure zero-touch (automated) transitive (network wide) IPv6 connectivity for all nodes in the same ACP domain as well as a GRASP instance running across this ACP IPv6 connectivity. The ACP is primarily meant to be used as a component of the ANI to enable Autonomic Networks but it can equally be used in simple ANI networks (with no other Autonomic Functions) or completely by itself.

ACP address: An IPv6 address assigned to the ACP node. It is stored in the domain information field of the ACP domain certificate.

ACP address range/set: The ACP address may imply a range or set of addresses that the node can assign for different purposes. This address range/set is derived by the node from the format of the ACP address called the addressing sub-scheme.

ACP connect interface: An interface on an ACP node providing access to the ACP for non ACP capable nodes without using an ACP secure channel. See Section 8.1.1.

ACP domain: The ACP domain is the set of nodes with ACP domain certificates that allow them to authenticate each other as members of the ACP domain. See also Section 6.1.2.
ACP (ANI/AN) Domain Certificate: A provisioned [RFC5280] certificate (LDevID) carrying the domain information field which is used by the ACP to learn its address in the ACP and to derive and cryptographically assert its membership in the ACP domain.

domain information (field): An rfc822Name information element (e.g., field) in the domain certificate in which the ACP relevant information is encoded: the domain name and the ACP address.

ACP Loopback interface: The Loopback interface in the ACP Virtual Routing and Forwarding (VRF) that has the ACP address assigned to it.

ACP network: The ACP network constitutes all the nodes that have access to the ACP. It is the set of active and transitively connected nodes of an ACP domain plus all nodes that get access to the ACP of that domain via ACP edge nodes.

ACP (ULA) prefix(es): The /48 IPv6 address prefixes used across the ACP. In the normal/simple case, the ACP has one ULA prefix, see Section 6.10. The ACP routing table may include multiple ULA prefixes if the "rsub" option is used to create addresses from more than one ULA prefix. See Section 6.1.1. The ACP may also include non-ULA prefixes if those are configured on ACP connect interfaces. See Section 8.1.1.

ACP secure channel: A cryptographically authenticated and encrypted data connection established between (normally) adjacent ACP nodes to carry traffic of the ACP VRF secure and isolated from Data-Plane traffic in-band over the same link/path as the Data-Plane.

ACP secure channel protocol: The protocol used to build an ACP secure channel, e.g., Internet Key Exchange Protocol version 2 (IKEv2) with IPsec or Datagram Transport Layer Security (DTLS).

ACP virtual interface: An interface in the ACP VRF mapped to one or more ACP secure channels. See Section 6.12.5.

AN "Autonomic Network": A network according to [I-D.ietf-anima-reference-model]. Its main components are ANI, Autonomic Functions and Intent.

(AN) Domain Name: An FQDN (Fully Qualified Domain Name) in the domain information field of the Domain Certificate. See Section 6.1.1.

ANI (nodes/network): "Autonomic Network Infrastructure". The ANI is the infrastructure to enable Autonomic Networks. It includes ACP,
BRSKI and GRASP. Every Autonomic Network includes the ANI, but not every ANI network needs to include autonomic functions beyond the ANI (nor Intent). An ANI network without further autonomic functions can for example support secure zero-touch (automated) bootstrap and stable connectivity for SDN networks – see [RFC8368].

ANIMA: "Autonomic Networking Integrated Model and Approach". ACP, BRSKI and GRASP are products of the IETF ANIMA working group.

ASA: "Autonomic Service Agent". Autonomic software modules running on an ANI device. The components making up the ANI (BRSKI, ACP, GRASP) are also described as ASAs.

Autonomic Function: A function/service in an Autonomic Network (AN) composed of one or more ASA across one or more ANI nodes.

BRSKI: "Bootstrapping Remote Secure Key Infrastructures" ([I-D.ietf-anima-bootstrapping-keyinfra]. A protocol extending EST to enable secure zero-touch bootstrap in conjunction with ACP. ANI nodes use ACP, BRSKI and GRASP.

Data-Plane: The counterpoint to the ACP VRF in an ACP node: all routing and forwarding in the node other than the ACP VRF. In a simple ACP or ANI node, the Data-Plane is typically provisioned by means other than autonomically, for example manually (including across the ACP) or via SDN controllers. In a fully Autonomic Network node, the Data-Plane is managed autonomically via Autonomic Functions and Intent. Note that other (non-ANIMA) RFCs use the Data-Plane to refer to what is better called the forwarding plane. This is not the way the term is used in this document!

device: A physical system, or physical node.

Enrollment: The process where a node presents identification (for example through keying material such as the private key of an IDevID) to a network and acquires a network specific identity and trust anchor such as an LDevID.

EST: "Enrollment over Secure Transport" ([RFC7030]). IETF standard-track protocol for enrollment of a node with an LDevID. BRSKI is based on EST.

GRASP: "Generic Autonomic Signaling Protocol". An extensible signaling protocol required by the ACP for ACP neighbor discovery.
The ACP also provides the "security and transport substrate" for the "ACP instance of GRASP". This instance of GRASP runs across the ACP secure channels to support BRSKI and other NOC/OAM or Autonomic Functions. See [I-D.ietf-anima-grasp].

IDevID: An "Initial Device IDentity" X.509 certificate installed by the vendor on new equipment. Contains information that establishes the identity of the node in the context of its vendor/manufacturer such as device model/type and serial number. See [AR8021]. IDevID cannot be used for the ACP because they are not provisioned by the owner of the network, so they can not directly indicate an ACP domain they belong to.

in-band (management): The type of management used predominantly in IP based networks, not leveraging an "out-of-band network". In in-band management, access to the managed equipment depends on the configuration of this equipment itself: interface, addressing, forwarding, routing, policy, security, management. This dependency makes in-band management fragile because the configuration actions performed may break in-band management connectivity. Breakage can not only be unintentional, it can simply be an unavoidable side effect of being unable to create configuration schemes where in-band management connectivity configuration is unaffected by Data-Plane configuration. See also "(virtual) out-of-band network".

Intent: Policy language of an autonomic network according to [I-D.ietf-anima-reference-model].

Loopback interface: The conventional name for an internal IP interface to which addresses may be assigned, but which transmits no external traffic.

LDevID: A "Local Device IDentity" is an X.509 certificate installed during "enrollment". The Domain Certificate used by the ACP is an LDevID. See [AR8021].

MIC: "Manufacturer Installed Certificate". Another word not used in this document to describe an IDevID.

native interface: Interfaces existing on a node without configuration of the already running node. On physical nodes these are usually physical interfaces. On virtual nodes their equivalent.

node: A system, e.g., supporting the ACP according to this document. Can be virtual or physical. Physical nodes are called devices.
Node-ID: The identifier of an ACP node inside that ACP. It is the last 64 (see Section 6.10.3) or 78-bits (see Section 6.10.5) of the ACP address.

Operational Technology (OT): "https://en.wikipedia.org/wiki/Operational_Technology" [1]: "The hardware and software dedicated to detecting or causing changes in physical processes through direct monitoring and/or control of physical devices such as valves, pumps, etc.". OT networks are today in most cases well separated from Information Technology (IT) networks.

(virtual) out-of-band network: An out-of-band network is a secondary network used to manage a primary network. The equipment of the primary network is connected to the out-of-band network via dedicated management ports on the primary network equipment. Serial (console) management ports were historically most common, higher end network equipment now also has ethernet ports dedicated only for management. An out-of-band network provides management access to the primary network independent of the configuration state of the primary network. One of the goals of the ACP is to provide this benefit of out-of-band networks virtually on the primary network equipment. The ACP VRF acts as a virtual out of band network device providing configuration independent management access. The ACP secure channels are the virtual links of the ACP virtual out-of-band network, meant to be operating independent of the configuration of the primary network. See also ->"in-band (management)" ()

RPL: "IPv6 Routing Protocol for Low-Power and Lossy Networks". The routing protocol used in the ACP. See [RFC6550].

MASA (service): "Manufacturer Authorized Signing Authority". A vendor/manufacturer or delegated cloud service on the Internet used as part of the BRSKI protocol.

(ACP/ANI/BRSKI) Registrar: An ACP registrar is an entity (software and/or person) that is orchestrating the enrollment of ACP nodes with the ACP domain certificate. ANI nodes use BRSKI, so ANI registrars are also called BRSKI registrars. For non-ANI ACP nodes, the registrar mechanisms are undefined by this document. See Section 6.10.7. Renewal and other maintenance (such as revocation) of ACP domain certificates may be performed by other entities than registrars. EST must be supported for ACP domain certificate renewal (see Section 6.1.4). BRSKI is an extension of EST, so ANI/BRSKI registrars can easily support ACP domain certificate renewal in addition to initial enrollment.
sUDI: "secured Unique Device Identifier", Another term not used in this document to refer to an IDevID.

UDI: "Unique Device Identifier", In the context of this document unsecured identity information of a node typically consisting of at least device model/type and serial number, often in a vendor specific format. See sUDI and LDevID.

ULA: (Global ID prefix) A "Unique Local Address" (ULA) is an IPv6 address in the block fc00::/7, defined in [RFC4193]. It is the approximate IPv6 counterpart of the IPv4 private address ([RFC1918]). The ULA Global ID prefix are the first 48-bits of a ULA address. In this document it is abbreviated as "ULA prefix".

(ACP) VRF: The ACP is modeled in this document as a "Virtual Routing and Forwarding" instance (VRF). This means that it is based on a "virtual router" consisting of a separate IPv6 forwarding table to which the ACP virtual interfaces are attached and an associated IPv6 routing table separate from the Data-Plane. Unlike the VRFs on MPLS/VPN-PE ([RFC4364]) or LISP XTR ([RFC6830]), the ACP VRF does not have any special "core facing" functionality or routing/mapping protocols shared across multiple VRFs. In vendor products a VRF such as the ACP-VRF may also be referred to as a so called VRF-lite.

(ACP) Zone: An ACP zone is a set of ACP nodes using the same zone field value in their ACP address according to Section 6.10.3. Zones are a mechanism to support structured addressing of ACP addresses within the same /48-bit ULA prefix.

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

### 3. Use Cases for an Autonomic Control Plane (Informative)

#### 3.1. An Infrastructure for Autonomic Functions

Autonomic Functions need a stable infrastructure to run on, and all autonomic functions should use the same infrastructure to minimize the complexity of the network. In this way, there is only need for a single discovery mechanism, a single security mechanism, and single instances of other processes that distributed functions require.
3.2. Secure Bootstrap over a not configured Network

Today, bootstrapping a new node typically requires all nodes between a controlling node such as an SDN controller ("Software Defined Networking", see [RFC7426]) and the new node to be completely and correctly addressed, configured and secured. Bootstrapping and configuration of a network happens in rings around the controller - configuring each ring of devices before the next one can be bootstrapped. Without console access (for example through an out-of-band network) it is not possible today to make devices securely reachable before having configured the entire network leading up to them.

With the ACP, secure bootstrap of new devices and whole new networks can happen without requiring any configuration of unconfigured devices along the path: As long as all devices along the path support ACP and a zero-touch bootstrap mechanism such as BRSKI, the ACP across a whole network of unconfigured devices can be brought up without operator/provisioning intervention. The ACP also provides additional security for any bootstrap mechanism, because it can provide encrypted discovery (via ACP GRASP) of registrars or other bootstrap servers by bootstrap proxies connecting to nodes that are to be bootstrapped and the ACP encryption hides the identities of the communicating entities (pledge and registrar), making it more difficult to learn which network node might be attackable. The ACP domain certificate can also be used to end-to-end encrypt the bootstrap communication between such proxies and server. Note that bootstrapping here includes not only the first step that can be provided by BRSKI (secure keys), but also later stages where configuration is bootstrapped.

3.3. Data-Plane Independent Permanent Reachability

Today, most critical control plane protocols and network management protocols are using the Data-Plane of the network. This leads to often undesirable dependencies between control and management plane on one side and the Data-Plane on the other: Only if the forwarding and control plane of the Data-Plane are configured correctly, will the Data-Plane and the management plane work as expected.

Data-Plane connectivity can be affected by errors and faults, for example misconfigurations that make AAA (Authentication, Authorization and Accounting) servers unreachable or can lock an administrator out of a device; routing or addressing issues can make a device unreachable; shutting down interfaces over which a current management session is running can lock an admin irreversibly out of the device. Traditionally only out-of-band access can help recover
from such issues (such as serial console or ethernet management port).

Data-Plane dependencies also affect applications in a Network Operations Center (NOC) such as SDN controller applications: Certain network changes are today hard to implement, because the change itself may affect reachability of the devices. Examples are address or mask changes, routing changes, or security policies. Today such changes require precise hop-by-hop planning.

Note that specific control plane functions for the Data-Plane often want to depend on forwarding of their packets via the Data-Plane: Aliveness and routing protocol signaling packets across the Data-Plane to verify reachability across the Data-Plane, using IPv4 signaling packets for IPv4 routing vs. IPv6 signaling packets for IPv6 routing.

Assuming appropriate implementation (see Section 6.12.2 for more details), the ACP provides reachability that is independent of the Data-Plane. This allows the control plane and management plane to operate more robustly:

- For management plane protocols, the ACP provides the functionality of a Virtual out-of-band (VooB) channel, by providing connectivity to all nodes regardless of their Data-Plane configuration, routing and forwarding tables.

- For control plane protocols, the ACP allows their operation even when the Data-Plane is temporarily faulty, or during transitional events, such as routing changes, which may affect the control plane at least temporarily. This is specifically important for autonomic service agents, which could affect Data-Plane connectivity.

The document "Using Autonomic Control Plane for Stable Connectivity of Network OAM" [RFC8368] explains this use case for the ACP in significantly more detail and explains how the ACP can be used in practical network operations.

4. Requirements (Informative)

The following requirements were identified for the design of the ACP based on the above use-cases (Section 3). These requirements are informative. The ACP as specified in the normative parts of this document is meeting or exceeding these use-case requirements:

ACP1: The ACP should provide robust connectivity: As far as possible, it should be independent of configured addressing,
configuration and routing. Requirements 2 and 3 build on this requirement, but also have value on their own.

ACP2: The ACP must have a separate address space from the Data-Plane. Reason: traceability, debug-ability, separation from Data-Plane, infrastructure security (filtering based on known address space).

ACP3: The ACP must use autonomically managed address space. Reason: easy bootstrap and setup ("autonomic"); robustness (admin cannot break network easily). This document suggests using ULA addressing for this purpose ("Unique Local Address", see [RFC4193]).

ACP4: The ACP must be generic, that is it must be usable by all the functions and protocols of the ANI. Clients of the ACP must not be tied to a particular application or transport protocol.

ACP5: The ACP must provide security: Messages coming through the ACP must be authenticated to be from a trusted node, and should (very strong should) be encrypted.

Explanation for ACP4: In a fully autonomic network (AN), newly written ASA could potentially all communicate exclusively via GRASP with each other, and if that was assumed to be the only requirement against the ACP, it would not need to provide IPv6 layer connectivity between nodes, but only GRASP connectivity. Nevertheless, because ACP also intends to support non-AN networks, it is crucial to support IPv6 layer connectivity across the ACP to support any transport and application layer protocols.

The ACP operates hop-by-hop, because this interaction can be built on IPv6 link local addressing, which is autonomic, and has no dependency on configuration (requirement 1). It may be necessary to have ACP connectivity across non-ACP nodes, for example to link ACP nodes over the general Internet. This is possible, but introduces a dependency against stable/resilient routing over the non-ACP hops (see Section 8.2).

5. Overview (Informative)

The Autonomic Control Plane is constructed in the following way (for details, see Section 6):

1. An ACP node creates a Virtual Routing and Forwarding (VRF) instance, or a similar virtual context.
2. It determines, following a policy, a candidate peer list. This is the list of nodes to which it should establish an Autonomic Control Plane. Default policy is: To all link-layer adjacent nodes supporting ACP.

3. For each node in the candidate peer list, it authenticates that node (according to Section 6.1.2) and negotiates a mutually acceptable channel type.

4. For each node in the candidate peer list, it then establishes a secure tunnel of the negotiated type. The resulting tunnels are then placed into the previously set up VRF. This creates an overlay network with hop-by-hop tunnels.

5. Inside the ACP VRF, each node assigns its ULA IPv6 address to a Loopback interface assigned to the ACP VRF.

6. Each node runs a lightweight routing protocol, to announce reachability of the virtual addresses inside the ACP (see Section 6.12.5).

Note:

- Non-autonomic NMS ("Network Management Systems") or SDN controllers have to be explicitly configured for connection into the ACP.

- Connecting over non-ACP Layer-3 clouds requires explicit configuration. See Section 8.2.

- None of the above operations (except explicit configured ones) are reflected in the configuration of the node.

The following figure illustrates the ACP.
6. Self-Creation of an Autonomic Control Plane (ACP) (Normative)

This section describes the components and steps to set up an Autonomic Control Plane (ACP), and highlights the key properties which make it "indestructible" against many inadvertent changes to the Data-Plane, for example caused by misconfigurations.

An ACP node can be a router, switch, controller, NMS host, or any other IP capable node. Initially, it must have it’s ACP domain certificate, as well as an (empty) ACP Adjacency Table (described in Section 6.2). It then can start to discover ACP neighbors and build the ACP. This is described step by step in the following sections:

6.1. ACP Domain, Certificate and Network

The ACP relies on group security. An ACP domain is a group of nodes that trust each other to participate in ACP operations. To establish trust, each ACP member requires keying material: An ACP node MUST have a certificate (LDevID) and a Trust Anchor (TA) consisting of a certificate (chain) used to sign the LDevID of all ACP domain members. The LDevID is used to cryptographically authenticate the membership of its owner node in the ACP domain to other ACP domain members, the TA is used to authenticate the ACP domain membership of other nodes (see Section 6.1.2).
The LDevID is called the ACP domain certificate, the TA is the Certificate Authority (CA) of the ACP domain.

The ACP does not mandate specific mechanisms by which this keying material is provisioned into the ACP node, it only requires the Domain information field as specified in Section 6.1.1 in its domain certificate as well as those of candidate ACP peers. See Appendix A.2 for more information about enrollment or provisioning options.

This document uses the term ACP in many places where the Autonomic Networking reference documents [RFC7575] and [I-D.ietf-anima-reference-model] use the word autonomic. This is done because those reference documents consider (only) fully autonomic networks and nodes, but support of ACP does not require support for other components of autonomic networks. Therefore the word autonomic might be misleading to operators interested in only the ACP.

[RFC7575] defines the term "Autonomic Domain" as a collection of autonomic nodes. ACP nodes do not need to be fully autonomic, but when they are, then the ACP domain is an autonomic domain. Likewise, [I-D.ietf-anima-reference-model] defines the term "Domain Certificate" as the certificate used in an autonomic domain. The ACP domain certificate is that domain certificate when ACP nodes are (fully) autonomic nodes. Finally, this document uses the term ACP network to refer to the network created by active ACP nodes in an ACP domain. The ACP network itself can extend beyond ACP nodes through the mechanisms described in Section 8.1.

The ACP domain certificate SHOULD be used for any authentication between nodes with ACP domain certificates (ACP nodes and NOC nodes) where the required condition is ACP domain membership, such as ACP node to NOC/OAM end-to-end security and ASA to ASA end-to-end security. Section 6.1.2 defines this "ACP domain membership check". The uses of this check that are standardized in this document are for the establishment of ACP secure channels (Section 6.6) and for ACP GRASP (Section 6.8.2).

6.1.1. Certificate ACP Domain Information Field

Information about the domain MUST be encoded in the domain certificate in a subjectAltName / rfc822Name field according to the following ABNF definition ([RFC5234]):

[RFC Editor: Please substitute SELF in all occurrences of rfcSELF in this document with the RFC number assigned to this document and remove this comment line]
domain-information = local-part "@" acp-domain-name
local-part = key [ "." local-info ]
key = "rfcSELF"
local-info = [ acp-address ] [ "+" rsub extensions ]
acp-address = 32hex-dig | 0
hex-dig = DIGIT / "a" / "b" / "c" / "d" / "e" / "f"

rsub = [ <subdomain> ] ; <subdomain> as of RFC1034, section 3.5
routing-subdomain = [ rsub "." ] acp-domain-name
acp-domain-name = ; <domain> ; as of RFC 1034, section 3.5
extensions = *( "+" extension )
extension = ; future standard definition.
; Must fit RFC5322 simple dot-atom format.

Example:
domain-information = rfcSELF+fd89b714f3db00000200000064000000
                      +area51.research@acp.example.com
acp-domain-name = acp.example.com
routing-subdomain = area51.research.acp.example.com

Figure 2: ACP Domain Information Field ABNF

Nodes complying with this specification MUST be able to receive their ACP address through the domain certificate, in which case their own ACP domain certificate MUST have the 32hex-dig "acp-address" field. Nodes complying with this specification MUST also be able to authenticate nodes as ACP domain members / ACP secure channel peers when they have an empty or 0-value acp-address field. See Section 6.1.2.

"acp-domain-name" is used to indicate the ACP Domain across which all ACP nodes trust each other and are willing to build ACP channels to each other. See Section 6.1.2. Acp-domain-name SHOULD be the FQDN of a DNS domain owned by the operator assigning the certificate. This is a simple method to ensure that the domain is globally unique and collision of ACP addresses would therefore only happen due to ULA hash collisions (see Section 6.10.2). If the operator does not own any FQDN, it should choose a string (in FQDN format) that it intends to be equally unique.

"routing-subdomain" is the autonomic subdomain composed of "rsub" and "acp-domain-name". "rsub" is optional. When not present, "routing-subdomain" is the same as "acp-domain-name". "routing-subdomain" determines the /48 ULA prefix for ACP addresses. "rsub" therefore allows to use multiple /48 ULA prefixes in an ACP domain. See Appendix A.7 for example use-cases.
The optional "extensions" field is used for future standardized extensions to this specification. It MUST be ignored if present and not understood.

Formatting notes:

- "rsub" needs to be in the "local-part": If the format just had routing-sub-domain as the domain part of the domain-information, rsub and acp-domain-name could not be separated from each other. It also makes acp-domain-name a valid e-mail target across all routing-subdomains.

- "acp-address" cannot use standard IPv6 address formats because it must match the simple dot-atom format of [RFC5322]. The character ":" is not allowed in that format.

- If "acp-address" is empty, and "rsub" is empty too, the "local-part" will have the format "rfcSELF++extension(s)". The two plus characters are necessary so the node can unambiguously parse that both "acp-address" and "rsub" are empty.

- The maximum size of "domain-information" is 254 characters and the maximum size of node-info is 64 characters according to [RFC5280] that is referring to [RFC2821] (superseded by [RFC5321]).

The subjectAltName / rfc822Name encoding of the ACP domain name and ACP address is used for the following reasons:

- It should be possible to share the LDevID with other uses beside the ACP. Therefore, the information element required for the ACP should be encoded so that it minimizes the possibility of creating incompatibilities with such other uses.

- The information for the ACP should not cause incompatibilities with any pre-existing ASN.1 software. This eliminates the introduction of a novel information element because that could require extensions to such pre-existing ASN.1 parsers.

- subjectAltName / rfc822Name is a pre-existing element that must be supported by all existing ASN.1 parsers for LDevID.

- The element required for the ACP should not be misinterpreted by any other uses of the LDevID. If the element used for the ACP is interpreted by other uses, the impact should be benign.

- The element should not require additional ASN.1 en/decoding, because it is unclear if all, especially embedded devices
certificate libraries would support extensible ASN.1 functionality.

- Using an IP address format encoding could result in non-benign misinterpretation of the domain information field; other uses unaware of the ACP could try to do something with the ACP address that would fail to work correctly. For example, the address could be interpreted to be an address of the node which does not belong to the ACP VRF.

- At minimum, both the AN domain name and the non-domain name derived part of the ACP address need to be encoded in one or more appropriate fields of the certificate, so there are not many alternatives with pre-existing fields where the only possible conflicts would likely be beneficial.

- rfc822Name encoding is very flexible. It allows to encode all the different fields of information required for the ACP.

- The format of the rfc822Name is chosen so that an operator can set up a mailbox called rfcSELF@<domain> that would receive emails sent towards the rfc822Name of any node inside a domain. This is possible because in many modern mail systems, components behind a "+" character are considered part of a single mailbox. In other words, it is not necessary to set up a separate mailbox for every ACP node, but only one for the whole domain.

- In result, if any unexpected use of the ACP addressing information in a certificate happens, it is benign and detectable: it would be mail to that mailbox.

See section 4.2.1.6 of [RFC5280] for details on the subjectAltName field.

6.1.2. ACP domain membership check

The following points constitute the ACP domain membership check of a candidate peer certificate, independent of the protocol used:

1: The peer certificate is valid (lifetime).

2: The peer has proved ownership of the private key associated with the certificate’s public key.

3: The peer’s certificate passes certificate path validation as defined in [RFC5280] against one of the Trust Anchors associated with the ACP nodes ACP domain certificate (see Section 6.1.3 below).
4: If the node certificate indicates a Certificate Revocation List (CRL) Distribution Point (CDP) ([RFC5280], section 4.2.1.13) or Online Certificate Status Protocol (OCSP) responder ([RFC5280], section 4.2.2.1), then the peer’s certificate must be valid according to those criteria: An OCSP check for the peer’s certificate across the ACP must succeed or the peer certificate must not be listed in the CRL retrieved from the CDP. This rule has to be skipped for ACP secure channel peer authentication when the node has no ACP or non-ACP connectivity to retrieve current CRL or access an OCSP responder (see below).

5: The peer’s certificate has a syntactically valid ACP domain information field (encoded as subjectAltName / rfc822Name) and the acp-domain-name in that peer’s domain information field is the same as in this ACP node’s certificate (lowercase normalized).

When an ACP node learns later via OCSP/CRL that an ACP peers certificate for which rule 4 had to be skipped during ACP secure channel establishment is invalid, then the ACP secure channel to that peer SHOULD be closed even if this peer is the only connectivity to access CRL/OCSP. The ACP secure channel connection MUST be retried periodically to support the case that the neighbor acquires a new, valid certificate.

Only when checking a candidate peer’s certificate for the purpose of establishing an ACP secure channel, one additional check is performed:

6: The candidate peer certificate’s ACP domain information field has a non-empty acp-address field (either 32hex-dig or 0, according to Figure 2).

Rule 6 for the establishment of ACP secure channels ensures that they will only be built between nodes which indicate through the acp-address in their ACP domain certificate the ability and permission by the Registrar to participate in ACP secure-channels.

Nodes with an empty acp-address field can only use their ACP domain certificate for non-ACP-secure channel authentication purposes.

The special value 0 in an ACP certificates acp-address field is used for nodes that can and should determine their ACP address through other mechanisms than learning it through the acp-address field in their ACP domain certificate. These ACP nodes are permitted to establish ACP secure channels. Mechanisms for those nodes to determine their ACP address are outside the scope of this specification.
Formally, the ACP domain membership check includes both the authentication of the peers certificate (steps 1...4) and a check authorizing this node and the peer to establish an ACP connection and/or any other secure connection across ACP or data-plane end to end. Step 5 authorizes to build any non-ACP secure connection between members of the same ACP domain, step 5 and 6 are required to build an ACP secure channel. For brevity, the remainder of this document refers to this process only as authentication instead of as authentication and authorization.

6.1.3. Trust Points and Trust Anchors

ACP nodes need Trust Point (TP) certificates to perform certificate path validation as required by Section 6.1.2, rule 3. Trust Point(s) must be provisioned to an ACP node (together with its ACP domain certificate) by an ACP Registrar during initial enrolment of a candidate ACP node. ACP nodes MUST also support renewal of TPs via EST as described below in Section 6.1.4.

Trust Point is the term used in this document for a certificate authority (CA) and its associated set of certificates. Multiple certificates are required for a CA to deal with CA certificate renewals as explained in Section 4.4 of CMP ([RFC4210]).

A certificate path is a chain of certificates starting at a self-signed certificate of a so called root-CA or Trust Anchor, followed by zero or more intermediate Trust Point or sub-CA certificates and ending with an ACP certificate. Certificate path validation authenticates that the ACP certificate is signed by a Trust Anchor, directly or indirectly via one or more intermediate Trust Points.

Note that different ACP nodes may have different Trust Points and even different Trust Anchors in their certificate path, as long as the set of Trust Points for all ACP node includes the same set of Trust Anchors (usually 1), and each ACP nodes set of Trust Anchors includes the intermediate Trust Points for its own ACP domain certificate. The protocols through which ACP domain membership check rules 1-4 are performed therefore need to support the exchange not only of the ACP nodes certificates, but also their intermediate Trust Points.

ACP nodes MUST support for the ACP domain membership check the certificate path validation with 0 or 1 intermediate Trust Points. They SHOULD support 2 intermediate Trust Points and two Trust Anchors (to permit migration to different root-CAs).

Trust Points for ACP domain certificates must be trusted to sign certificates with valid ACP domain information fields only for
trusted ACP registrars of that domain. This can be achieved by using Trust Anchors private to the owner of the ACP domain or potentially through appropriate contractual agreements between the involved parties. Public CA without such obligations and guarantees can not be used.

A single owner can operate multiple independent ACP domains from the same set of private trust anchors (CAs) when the ACP Registrars are trusted not to permit certificates with incorrect ACP information fields to be signed. Such as ACP information with a wrong acp-domain field. In this case, CAs can be completely unaware of ACP specifics, so that it should be possible to use any existing CA software. When ACP Registrars are not to be trusted, the correctness of the ACP domain information field for the candidate ACP node has to be verified by the CA signing the ACP domain certificate.

6.1.4. Certificate and Trust Point Maintenance

ACP nodes MUST support renewal of their Certificate and Trust Points (TP) via EST ("Enrollment over Secure Transport", see [RFC7030]) and MAY support other mechanisms. An ACP network MUST have at least one ACP node supporting EST server functionality across the ACP so that EST renewal is useable.

ACP nodes SHOULD be able to remember the EST server from which they last renewed their ACP domain certificate and SHOULD provide the ability for this remembered EST server to also be set by the ACP Registrar (see Section 6.10.7) that initially enrolled the ACP device with its ACP domain certificate. When BRSKI (see [I-D.ietf-anima-bootstrapping-keyinfra]) is used, the ACP address of the BRSKI registrar from the BRSKI TLS connection SHOULD be remembered and used for the next renewal via EST if that registrar also announces itself as an EST server via GRASP (see next section) on its ACP address.

6.1.4.1. GRASP objective for EST server

ACP nodes that are EST servers MUST announce their service via GRASP in the ACP through M_FLOOD messages. See [I-D.ietf-anima-grasp], section 2.8.11 for the definition of this message type:
Example:

```
[M_FLOOD, 12340815, h'fd89b714f3db000020000064000001', 210000,
  ["SRV.est", 4, 255 ],
  [O_IPv6_LOCATOR,
    h'fd89b714f3db000020000064000001', TCP, 80]
]
```

Figure 3: GRASP SRV.est example

The formal definition of the objective in Concise data definition language (CDDL) (see [I-D.ietf-cbor-cddl]) is as follows:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
  +[objective, (locator-option / [])]]
objective = ["SRV.est", objective-flags, loop-count,
  objective-value]
objective-flags = sync-only  ; as in GRASP spec
sync-only = 4          ; M_FLOOD only requires synchronization
loop-count = 255        ; recommended
objective-value = any    ; Not used (yet)
```

Figure 4: GRASP SRV.est definition

The objective name "SRV.est" indicates that the objective is an [RFC7030] compliant EST server because "est" is an [RFC6335] registered service name for [RFC7030]. Objective-value MUST be ignored if present. Backward compatible extensions to [RFC7030] MAY be indicated through objective-value. Non [RFC7030] compatible certificate renewal options MUST use a different objective-name.

The M_FLOOD message MUST be sent periodically. The default SHOULD be 60 seconds, the value SHOULD be operator configurable but SHOULD be not smaller than 60 seconds. The frequency of sending MUST be such that the aggregate amount of periodic M_FLOODs from all flooding sources cause only negligible traffic across the ACP. The time-to-live (ttl) parameter SHOULD be 3.5 times the period so that up to three consecutive messages can be dropped before considering an announcement expired. In the example above, the ttl is 210000 msec, 3.5 times 60 seconds. When a service announcer using these parameters unexpectedly dies immediately after sending the M_FLOOD, receivers would consider it expired 210 seconds later. When a receiver tries to connect to this dead service before this timeout, it will experience a failing connection and use that as an indication
that the service is dead and select another instance of the same service instead.

6.1.4.2. Renewal

When performing renewal, the node SHOULD attempt to connect to the remembered EST server. If that fails, it SHOULD attempt to connect to an EST server learned via GRASP. The server with which certificate renewal succeeds SHOULD be remembered for the next renewal.

Remembering the last renewal server and preferring it provides stickiness which can help diagnostics. It also provides some protection against off-path compromised ACP members announcing bogus information into GRASP.

Renewal of certificates SHOULD start after less than 50% of the domain certificate lifetime so that network operations has ample time to investigate and resolve any problems that causes a node to not renew its domain certificate in time - and to allow prolonged periods of running parts of a network disconnected from any CA.

6.1.4.3. Certificate Revocation Lists (CRLs)

The ACP node SHOULD support Certificate Revocation Lists (CRL) via HTTPs from one or more CRL Distribution Points (CDPs). The CDP(s) MUST be indicated in the Domain Certificate when used. If the CDP URL uses an IPv6 address (ULA address when using the addressing rules specified in this document), the ACP node will connect to the CDP via the ACP. If the CDP uses a domain name, the ACP node will connect to the CDP via the Data-Plane.

It is common to use domain names for CDP(s), but there is no requirement for the ACP to support DNS. Any DNS lookup in the Data-Plane is not only a possible security issue, but it would also not indicate whether the resolved address is meant to be reachable across the ACP. Therefore, the use of an IPv6 address versus the use of a DNS name doubles as an indicator whether or not to reach the CDP via the ACP.

A CDP can be reachable across the ACP either by running it on a node with ACP or by connecting its node via an ACP connect interface (see Section 8.1). The CDP SHOULD use an ACP domain certificate for its HTTPs connections. The connecting ACP node SHOULD verify that the CDP certificate used during the HTTPs connection has the same ACP address as indicated in the CDP URL of the nodes ACP domain certificate if the CDP URL uses an IPv6 address.
6.1.4.4. Lifetimes

Certificate lifetime may be set to shorter lifetimes than customary (1 year) because certificate renewal is fully automated via ACP and EST. The primary limiting factor for shorter certificate lifetimes is load on the EST server(s) and CA. It is therefore recommended that ACP domain certificates are managed via a CA chain where the assigning CA has enough performance to manage short lived certificates. See also Section 10.2.4 for discussion about an example setup achieving this. See also [I-D.ietf-acme-star].

When certificate lifetimes are sufficiently short, such as few hours, certificate revocation may not be necessary, allowing to simplify the overall certificate maintenance infrastructure.

See Appendix A.2 for further optimizations of certificate maintenance when BRSKI can be used ("Bootstrapping Remote Secure Key Infrastructures", see [I-D.ietf-anima-bootstrapping-keyinfra]).

6.1.4.5. Re-enrollment

An ACP node may determine that its ACP domain certificate has expired, for example because the ACP node was powered down or disconnected longer than its certificate lifetime. In this case, the ACP node SHOULD convert to a role of a re-enrolling candidate ACP node.

In this role, the node does maintain the trust anchor and certificate chain associated with its ACP domain certificate exclusively for the purpose of re-enrollment, and attempts (or waits) to get re-enrolled with a new ACP certificate. The details depend on the mechanisms/protocols used by the ACP registrars.

Please refer to Section 6.10.7 and [I-D.ietf-anima-bootstrapping-keyinfra] for explanations about ACP registrars and vouchers as used in the following text. When ACP is intended to be used without BRSKI, the details about BRSKI and vouchers in the following text can be skipped.

When BRSKI is used (i.e.: on ACP nodes that are ANI nodes), the re-enrolling candidate ACP node would attempt to enroll like a candidate ACP node (BRSKI pledge), but instead of using the ACP nodes IDevID, it SHOULD first attempt to use its ACP domain certificate in the BRSKI TLS authentication. The BRSKI registrar MAY honor this certificate beyond its expiration date purely for the purpose of re-enrollment. Using the ACP node’s domain certificate allows the BRSKI registrar to learn that nodes ACP domain information field, so that
the BRSKI registrar can re-assign the same ACP address information to
the ACP node in the new ACP domain certificate.

If the BRSKI registrar denies the use of the old ACP domain
certificate, the re-enrolling candidate ACP node MUST re-attempt re-
enrollment using its IDevID as defined in BRSKI during the TLS
connection setup.

Both when the BRSKI connection is attempted with the old ACP domain
certificate or the IDevID, the re-enrolling candidate ACP node SHOULD
authenticate the BRSKI registrar during TLS connection setup based on
its existing trust anchor/certificate chain information associated
with its old ACP certificate. The re-enrolling candidate ACP node
SHOULD only request a voucher from the BRSKI registrar when this
authentication fails during TLS connection setup.

When other mechanisms than BRSKI are used for ACP domain certificate
enrollment, the principles of the re-enrolling candidate ACP node are
the same. The re-enrolling candidate ACP node attempts to
authenticate any ACP registrar peers during re-enrollment protocol/
mechanisms via its existing certificate chain/trust anchor and
provides its existing ACP domain certificate and other identification
(such as the IDevID) as necessary to the registrar.

Maintaining existing trust anchor information is especially important
when enrollment mechanisms are used that unlike BRSKI do not leverage
a voucher mechanism to authenticate the ACP registrar and where
therefore the injection of certificate failures could otherwise make
the ACP node easily attackable remotely.

When using BRSKI or other protocol/mechanisms supporting vouchers,
maintaining existing trust anchor information allows for re-
enrollment of expired ACP certificates to be more lightweight,
especially in environments where repeated acquisition of vouchers
during the lifetime of ACP nodes may be operationally expensive or
otherwise undesirable.

6.1.4.6.  Failing Certificates

An ACP domain certificate is called failing in this document, if/when
the ACP node can determine that it was revoked (or explicitly not
renewed), or in the absence of such explicit local diagnostics, when
the ACP node fails to connect to other ACP nodes in the same ACP
domain using its ACP certificate. For connection failures to
determine the ACP domain certificate as the culprit, the peer should
pass the domain membership check (Section 6.1.2) and other reasons
for the connection failure can be excluded because of the connection
error diagnostics.
This type of failure can happen during setup/refresh of a secure ACP channel connections or any other use of the ACP domain certificate, such as for the TLS connection to an EST server for the renewal of the ACP domain certificate.

Example reasons for failing certificates that the ACP node can only discover through connection failure are that the domain certificate or any of its signing certificates could have been revoked or may have expired, but the ACP node cannot self-diagnose this condition directly. Revocation information or clock synchronization may only be available across the ACP, but the ACP node cannot build ACP secure channels because ACP peers reject the ACP node’s domain certificate.

ACP nodes SHOULD support the option to determine whether its ACP certificate is failing, and when it does, put itself into the role of a re-enrolling candidate ACP node as explained above (Section 6.1.4.5).

6.2. ACP Adjacency Table

To know to which nodes to establish an ACP channel, every ACP node maintains an adjacency table. The adjacency table contains information about adjacent ACP nodes, at a minimum: Node-ID (identifier of the node inside the ACP, see Section 6.10.3 and Section 6.10.5), interface on which neighbor was discovered (by GRASP as explained below), link-local IPv6 address of neighbor on that interface, certificate (including domain information field). An ACP node MUST maintain this adjacency table. This table is used to determine to which neighbor an ACP connection is established.

Where the next ACP node is not directly adjacent (i.e., not on a link connected to this node), the information in the adjacency table can be supplemented by configuration. For example, the Node-ID and IP address could be configured. See Section 8.2.

The adjacency table MAY contain information about the validity and trust of the adjacent ACP node’s certificate. However, subsequent steps MUST always start with the ACP domain membership check against the peer (see Section 6.1.2).

The adjacency table contains information about adjacent ACP nodes in general, independently of their domain and trust status. The next step determines to which of those ACP nodes an ACP connection should be established.
6.3. Neighbor Discovery with DULL GRASP

[ RFC Editor: GRASP draft is in RFC editor queue, waiting for dependencies, including ACP. Please ensure that references to I-D.ietf-anima-grasp that include section number references (throughout this document) will be updated in case any last-minute changes in GRASP would make those section references change.]

Discovery Unsolicited Link-Local (DULL) GRASP is a limited subset of GRASP intended to operate across an insecure link-local scope. See section 2.5.2 of [I-D.ietf-anima-grasp] for its formal definition. The ACP uses one instance of DULL GRASP for every L2 interface of the ACP node to discover link level adjacent candidate ACP neighbors. Unless modified by policy as noted earlier (Section 5 bullet point 2.), native interfaces (e.g., physical interfaces on physical nodes) SHOULD be initialized automatically to a state in which ACP discovery can be performed and any native interfaces with ACP neighbors can then be brought into the ACP even if the interface is otherwise not configured. Reception of packets on such otherwise not configured interfaces MUST be limited so that at first only IPv6 Stateless Address Auto Configuration (SLAAC - [RFC4862]) and DULL GRASP work and then only the following ACP secure channel setup packets - but not any other unnecessary traffic (e.g., no other link-local IPv6 transport stack responders for example).

Note that the use of the IPv6 link-local multicast address (ALL_GRASP_NEIGHBORS) implies the need to use Multicast Listener Discovery Version 2 (MLDv2, see [RFC3810]) to announce the desire to receive packets for that address. Otherwise DULL GRASP could fail to operate correctly in the presence of MLD snooping, non-ACP enabled L2 switches - because those would stop forwarding DULL GRASP packets. Switches not supporting MLD snooping simply need to operate as pure L2 bridges for IPv6 multicast packets for DULL GRASP to work.

ACP discovery SHOULD NOT be enabled by default on non-native interfaces. In particular, ACP discovery MUST NOT run inside the ACP across ACP virtual interfaces. See Section 10.3 for further, non-normative suggestions on how to enable/disable ACP at node and interface level. See Section 8.2.2 for more details about tunnels (typical non-native interfaces). See Section 7 for how ACP should be extended on devices operating (also) as L2 bridges.

Note: If an ACP node also implements BRSKI to enroll its ACP domain certificate (see Appendix A.2 for a summary), then the above considerations also apply to GRASP discovery for BRSKI. Each DULL instance of GRASP set up for ACP is then also used for the discovery of a bootstrap proxy via BRSKI when the node does not have a domain certificate. Discovery of ACP neighbors happens only when the node...
does have the certificate. The node therefore never needs to
discover both a bootstrap proxy and ACP neighbor at the same time.

An ACP node announces itself to potential ACP peers by use of the
"AN_ACP" objective. This is a synchronization objective intended to
be flooded on a single link using the GRASP Flood Synchronization
(M_FLOOD) message. In accordance with the design of the Flood
message, a locator consisting of a specific link-local IP address, IP
protocol number and port number will be distributed with the flooded
objective. An example of the message is informally:

```
[M_FLOOD, 12340815, h’fe80000000000000c0011001FEEF0000, 210000,
 ["AN_ACP", 4, 1, "IKEv2" ],
 [O_IPv6_LOCATOR,
  h’fe80000000000000c0011001FEEF0000, UDP, 15000]
 ["AN_ACP", 4, 1, "DTLS" ],
 [O_IPv6_LOCATOR,
  h’fe80000000000000c0011001FEEF0000, UDP, 17000]
 ]
```

Figure 5: GRASP AN_ACP example

The formal CDDL definition is:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
 +[objective, (locator-option / [])]]

objective = ["AN_ACP", objective-flags, loop-count,
             objective-value]

objective-flags = sync-only ; as in the GRASP specification
sync-only = 4    ; M_FLOOD only requires synchronization
loop-count = 1    ; limit to link-local operation
objective-value = method
method = "IKEv2" / "DTLS"  ; or future standard methods
```

Figure 6: GRASP AN_ACP definition

The objective-flags field is set to indicate synchronization.
The loop-count is fixed at 1 since this is a link-local operation.

In the above example the RECOMMENDED period of sending of the
objective is 60 seconds. The indicated ttl of 210000 msec means that
the objective would be cached by ACP nodes even when two out of three
messages are dropped in transit.
The session-id is a random number used for loop prevention (distinguishing a message from a prior instance of the same message). In DULL this field is irrelevant but must still be set according to the GRASP specification.

The originator MUST be the IPv6 link local address of the originating ACP node on the sending interface.

The 'objective-value' parameter is a string indicating the secure channel protocol available at the specified or implied locator.

The locator-option is optional and only required when the secure channel protocol is not offered at a well-defined port number, or if there is no well-defined port number.

"IKEv2" is the abbreviation for "Internet Key Exchange protocol version 2". It is the main protocol used by the Internet IP security architecture (IPsec). We therefore use the term "IKEv2" and not "IPsec" in the GRASP definitions and example above. "IKEv2" has a well-defined port number 500, but in the above example, the candidate ACP neighbor is offering ACP secure channel negotiation via IKEv2 on port 15000 (for the sake of creating a non-standard example).

"DTLS" indicates datagram Transport Layer Security version 1.2. There is no default UDP port, it must always be locally assigned by the node. See Section 6.7.2.

If a locator is included, it MUST be an O_IPv6_LOCATOR, and the IPv6 address MUST be the same as the initiator address (these are DULL requirements to minimize third party DoS attacks).

The secure channel methods defined in this document use the objective-values of "IKEv2" and "DTLS". There is no distinction between IKEv2 native and GRE-IKEv2 because this is purely negotiated via IKEv2.

A node that supports more than one secure channel protocol method needs to flood multiple versions of the "AN_ACP" objective so that each method can be accompanied by its own locator-option. This can use a single GRASP M_FLOOD message as shown in Figure 5.

Note that a node serving both as an ACP node and BRSKI Join Proxy may choose to distribute the "AN_ACP" objective and the respective BRSKI in the same M_FLOOD message, since GRASP allows multiple objectives in one message. This may be impractical though if ACP and BRSKI operations are implemented via separate software modules / ASAs.
The result of the discovery is the IPv6 link-local address of the neighbor as well as its supported secure channel protocols (and non-standard port they are running on). It is stored in the ACP Adjacency Table (see Section 6.2), which then drives the further building of the ACP to that neighbor.

6.4. Candidate ACP Neighbor Selection

An ACP node must determine to which other ACP nodes in the adjacency table it should build an ACP connection. This is based on the information in the ACP Adjacency table.

The ACP is established exclusively between nodes in the same domain. This includes all routing subdomains. Appendix A.7 explains how ACP connections across multiple routing subdomains are special.

The result of the candidate ACP neighbor selection process is a list of adjacent or configured autonomic neighbors to which an ACP channel should be established. The next step begins that channel establishment.

6.5. Channel Selection

To avoid attacks, initial discovery of candidate ACP peers cannot include any non-protected negotiation. To avoid re-inventing and validating security association mechanisms, the next step after discovering the address of a candidate neighbor can only be to try first to establish a security association with that neighbor using a well-known security association method.

At this time in the lifecycle of ACP nodes, it is unclear whether it is feasible to even decide on a single MTI (mandatory to implement) security association protocol across all ACP nodes.

From the use-cases it seems clear that not all type of ACP nodes can or need to connect directly to each other or are able to support or prefer all possible mechanisms. For example, code space limited IoT devices may only support DTLS because that code exists already on them for end-to-end security, but low-end in-ceiling L2 switches may only want to support Media Access Control Security (MacSec, see 802.1AE ([MACSEC])) because that is also supported in their chips. Only a flexible gateway device may need to support both of these mechanisms and potentially more. Note that MacSec is not required by any profiles of the ACP in this specification but just mentioned as a likely next interesting secure channel protocol.

To support extensible secure channel protocol selection without a single common MTI protocol, ACP nodes must try all the ACP secure
channel protocols it supports and that are feasible because the candidate ACP neighbor also announced them via its AN_ACP GRASP parameters (these are called the "feasible" ACP secure channel protocols).

To ensure that the selection of the secure channel protocols always succeeds in a predictable fashion without blocking, the following rules apply:

- An ACP node may choose to attempt to initiate the different feasible ACP secure channel protocols it supports according to its local policies sequentially or in parallel, but it MUST support acting as a responder to all of them in parallel.

- Once the first secure channel protocol succeeds, the two peers know each other’s certificates because they must be used by all secure channel protocols for mutual authentication. The node with the lower Node-ID in the ACP address becomes Bob, the one with the higher Node-ID in the certificate Alice.

- Bob becomes passive, he does not attempt to further initiate ACP secure channel protocols with Alice and does not consider it to be an error when Alice closes secure channels. Alice becomes the active party, continues to attempt setting up secure channel protocols with Bob until she arrives at the best one from her view that also works with Bob.

For example, originally Bob could have been the initiator of one ACP secure channel protocol that Bob prefers and the security association succeeded. The roles of Bob and Alice are then assigned and the connection setup is completed. The protocol could for example be IPsec via IKEv2 ("IP security", see [RFC4301] and "Internet Key Exchange protocol version 2", see [RFC7296]). It is now up to Alice to decide how to proceed. Even if the IPsec connection from Bob succeeded, Alice might prefer another secure protocol over IPsec (e.g., FOOBAR), and try to set that up with Bob. If that preference of Alice succeeds, she would close the IPsec connection. If no better protocol attempt succeeds, she would keep the IPsec connection.
The following sequence of steps show this example in more detail:

[1] Node 1 sends GRASP AN_ACP message to announce itself

[2] Node 2 sends GRASP AN_ACP message to announce itself


[4:C1] Because of [3], Node 2 starts as initiator on its preferred secure channel protocol towards Node 1. Connection C1.


[6:C2] Because of [5], Node 1 starts as initiator on its preferred secure channel protocol towards Node 2. Connection C2.

[7:C1] Node1 and Node2 have authenticated each others certificate on connection C1 as valid ACP peers.

[8:C1] Node 1 certificate has lower ACP Node-ID than Node2, therefore Node 1 considers itself Bob and Node 2 Alice on connection C1. Connection setup C1 is completed.

[9] Node 1 (Bob) refrains from attempting any further secure channel connections to Node 2 (Alice) as learned from [2] because it knows from [8:C1] that it is Bob relative to Node 1.

[10:C2] Node1 and Node2 have authenticated each others certificate on connection C2 (like [7:C1]).

[11:C2] Node 2 certificate has lower ACP Node-ID than Node2, therefore Node 1 considers itself Bob and Node 2 Alice on connection C1, but they also identify that C2 is to the same mutual peer as their C1, so this has no further impact.

[12:C2] Node 1 (Alice) closes C1. Because of [8:C1], Node 2 (Bob) expected this.

[13] Node 1 (Alice) and Node 2 (Bob) start data transfer across C2, which makes it become a secure channel for the ACP.

Figure 7: Secure Channel sequence of steps

All this negotiation is in the context of an "L2 interface". Alice and Bob will build ACP connections to each other on every "L2
An autonomic node must not assume that neighbors with the same L2 or link-local IPv6 addresses on different L2 interfaces are the same node. This can only be determined after examining the certificate after a successful security association attempt.

6.6. Candidate ACP Neighbor verification

Independent of the security association protocol chosen, candidate ACP neighbors need to be authenticated based on their domain certificate. This implies that any secure channel protocol MUST support certificate based authentication that can support the ACP domain membership check as defined in Section 6.1.2. If it fails, the connection attempt is aborted and an error logged. Attempts to reconnect MUST be throttled. The RECOMMENDED default is exponential base 2 backoff with a minimum delay of 10 seconds and a maximum delay of 640 seconds.

6.7. Security Association protocols

The following sections define the security association protocols that we consider to be important and feasible to specify in this document:

6.7.1. ACP via IKEv2

An ACP node announces its ability to support IKEv2 as the ACP secure channel protocol in GRASP as "IKEv2".

6.7.1.1. Native IPsec

To run ACP via IPsec natively, no further IANA assignments/definitions are required. An ACP node that is supporting native IPsec MUST use IPsec security setup via IKEv2, tunnel mode, local and peer link-local IPv6 addresses used for encapsulation. It MUST then support ESP with AES-256-GCM ([RFC4106]) for encryption and SHA256 hash and MUST NOT permit weaker crypto options. Key establishment MUST support ECDHE with P-256.

In terms of IKEv2, this means the initiator will offer to support IPsec tunnel mode with next protocol equal to 41 (IPv6).

IPsec tunnel mode is required because the ACP will route/forward packets received from any other ACP node across the ACP secure channels, and not only its own generated ACP packets. With IPsec transport mode, it would only be possible to send packets originated by the ACP node itself.
ESP is used because ACP mandates the use of encryption for ACP secure channels.

6.7.1.2. IPsec with GRE encapsulation

In network devices it is often more common to implement high performance virtual interfaces on top of GRE encapsulation than on top of a "native" IPsec association (without any other encapsulation than those defined by IPsec). On those devices it may be beneficial to run the ACP secure channel on top of GRE protected by the IPsec association.

To run ACP via GRE/IPsec, no further IANA assignments/definitions are required. An ACP node that is supporting ACP via GRE/IPsec MUST then support IPsec security setup via IKEv2, IPsec transport mode, local and peer link-local IPv6 addresses used for encapsulation, ESP with AES256 encryption and SHA256 hash.

When GRE is used, transport mode is sufficient because the routed ACP packets are not "tunneled" by IPsec but rather by GRE: IPsec only has to deal with the GRE/IP packet which always uses the local and peer link-local IPv6 addresses and is therefore applicable to transport mode.

ESP is used because ACP mandates the use of encryption for ACP secure channels.

In terms of IKEv2 negotiation, this means the initiator must offer to support IPsec transport mode with next protocol equal to GRE (47) followed by the offer for native IPsec as described above (because that option is mandatory to support).

If IKEv2 initiator and responder support GRE, it will be selected. The version of GRE to be used must be determined according to [RFC7676].

6.7.2. ACP via DTLS

We define the use of ACP via DTLS in the assumption that it is likely the first transport encryption code basis supported in some classes of constrained devices.

To run ACP via UDP and DTLS v1.2 [RFC6347] a locally assigned UDP port is used that is announced as a parameter in the GRASP AN_ACP objective to candidate neighbors.

All ACP nodes supporting DTLS as a secure channel protocol MUST adhere to the DTLS implementation recommendations and security
considerations of [RFC7525] except with respect to the DTLS version. ACP nodes supporting DTLS MUST implement only DTLS 1.2 or later. For example, implementing DTLS-1.3 ([I-D.ietf-tls-dtls13]) is also an option.

There is no additional session setup or other security association besides this simple DTLS setup. As soon as the DTLS session is functional, the ACP peers will exchange ACP IPv6 packets as the payload of the DTLS transport connection. Any DTLS defined security association mechanisms such as re-keying are used as they would be for any transport application relying solely on DTLS.

6.7.3. ACP Secure Channel Requirements

As explained in the beginning of Section 6.5, there is no single secure channel mechanism mandated for all ACP nodes. Instead, this section defines two ACP profiles (baseline and constrained) for ACP nodes that do introduce such requirements.

A baseline ACP node MUST support IPsec natively and MAY support IPsec via GRE. A constrained ACP node that cannot support IPsec MUST support DTLS. An ACP node connecting an area of constrained ACP nodes with an area of baseline ACP nodes MUST therefore support IPsec and DTLS and supports therefore the baseline and constrained profile.

Explanation: Not all type of ACP nodes can or need to connect directly to each other or are able to support or prefer all possible secure channel mechanisms. For example, code space limited IoT devices may only support DTLS because that code exists already on them for end-to-end security, but high-end core routers may not want to support DTLS because they can perform IPsec in accelerated hardware but would need to support DTLS in an underpowered CPU forwarding path shared with critical control plane operations. This is not a deployment issue for a single ACP across these type of nodes as long as there are also appropriate gateway ACP nodes that support sufficiently many secure channel mechanisms to allow interconnecting areas of ACP nodes with a more constrained set of secure channel protocols. On the edge between IoT areas and high-end core networks, general-purpose routers that act as those gateways and that can support a variety of secure channel protocols is the norm already.

ACP nodes need to specify in documentation the set of secure ACP mechanisms they support and should declare which profile they support according to above requirements.

An ACP secure channel MUST immediately be terminated when the lifetime of any certificate in the chain used to authenticate the neighbor expires or becomes revoked. Note that this is not standard
behavior in secure channel protocols such as IPsec because the certificate authentication only influences the setup of the secure channel in these protocols.

6.8. GRASP in the ACP

6.8.1. GRASP as a core service of the ACP

The ACP MUST run an instance of GRASP inside of it. It is a key part of the ACP services. The function in GRASP that makes it fundamental as a service of the ACP is the ability to provide ACP wide service discovery (using objectives in GRASP).

ACP provides IP unicast routing via the RPL routing protocol (see Section 6.11).

The ACP does not use IP multicast routing nor does it provide generic IP multicast services (the handling of GRASP link-local multicast messages is explained in Section 6.8.2). Instead, the ACP provides service discovery via the objective discovery/announcement and negotiation mechanisms of the ACP GRASP instance (services are a form of objectives). These mechanisms use hop-by-hop reliable flooding of GRASP messages for both service discovery (GRASP M_DISCOVERY messages) and service announcement (GRASP M_FLOOD messages).

See Appendix A.5 for discussion about this design choice of the ACP.

6.8.2. ACP as the Security and Transport substrate for GRASP

In the terminology of GRASP ([I-D.ietf-anima-grasp]), the ACP is the security and transport substrate for the GRASP instance run inside the ACP ("ACP GRASP").

This means that the ACP is responsible for ensuring that this instance of GRASP is only sending messages across the ACP GRASP virtual interfaces. Whenever the ACP adds or deletes such an interface because of new ACP secure channels or loss thereof, the ACP needs to indicate this to the ACP instance of GRASP. The ACP exists also in the absence of any active ACP neighbors. It is created when the node has a domain certificate, and continues to exist even if all of its neighbors cease operation.

In this case ASAs using GRASP running on the same node would still need to be able to discover each other’s objectives. When the ACP does not exist, ASAs leveraging the ACP instance of GRASP via APIs MUST still be able to operate, and MUST be able to understand that there is no ACP and that therefore the ACP instance of GRASP cannot operate.
The way ACP acts as the security and transport substrate for GRASP is visualized in the following picture:
Figure 8: ACP as security and transport substrate for GRASP
GRASP unicast messages inside the ACP always use the ACP address. Link-local addresses from the ACP VRF must not be used inside objectives. GRASP unicast messages inside the ACP are transported via TLS 1.2 ([RFC5246]) connections with AES256 encryption and SHA256. Mutual authentication uses the ACP domain membership check defined in (Section 6.1.2).

GRASP link-local multicast messages are targeted for a specific ACP virtual interface (as defined Section 6.12.5) but are sent by the ACP into an ACP GRASP virtual interface that is constructed from the TCP connection(s) to the IPv6 link-local neighbor address(es) on the underlying ACP virtual interface. If the ACP GRASP virtual interface has two or more neighbors, the GRASP link-local multicast messages are replicated to all neighbor TCP connections.

TCP and TLS connections for GRASP in the ACP use the IANA assigned TCP port for GRASP (7107). Effectively the transport stack is expected to be TLS for connections from/to the ACP address (e.g., global scope address(es)) and TCP for connections from/to link-local addresses on the ACP virtual interfaces. The latter ones are only used for flooding of GRASP messages.

6.8.2.1. Discussion

TCP encapsulation for GRASP M_DISCOVERY and M_FLOOD link local messages is used because these messages are flooded across potentially many hops to all ACP nodes and a single link with even temporary packet loss issues (e.g., WiFi/Powerline link) can reduce the probability for loss free transmission so much that applications would want to increase the frequency with which they send these messages. Such shorter periodic retransmission of datagrams would result in more traffic and processing overhead in the ACP than the hop-by-hop reliable retransmission mechanism by TCP and duplicate elimination by GRASP.

TLS is mandated for GRASP non-link-local unicast because the ACP secure channel mandatory authentication and encryption protects only against attacks from the outside but not against attacks from the inside: Compromised ACP members that have (not yet) been detected and removed (e.g., via domain certificate revocation / expiry).

If GRASP peer connections would just use TCP, compromised ACP members could simply eavesdrop passively on GRASP peer connections for whom they are on-path ("Man In The Middle" - MITM). Or intercept and modify them. With TLS, it is not possible to completely eliminate problems with compromised ACP members, but attacks are a lot more complex:
Eavesdropping/spoofing by a compromised ACP node is still possible because in the model of the ACP and GRASP, the provider and consumer of an objective have initially no unique information (such as an identity) about the other side which would allow them to distinguish a benevolent from a compromised peer. The compromised ACP node would simply announce the objective as well, potentially filter the original objective in GRASP when it is a MITM and act as an application level proxy. This of course requires that the compromised ACP node understand the semantics of the GRASP negotiation to an extent that allows it to proxy it without being detected, but in an ACP environment this is quite likely public knowledge or even standardized.

The GRASP TLS connections are run the same as any other ACP traffic through the ACP secure channels. This leads to double authentication/encryption, which has the following benefits:

- Secure channel methods such as IPsec may provide protection against additional attacks, for example reset-attacks.
- The secure channel method may leverage hardware acceleration and there may be little or no gain in eliminating it.
- There is no different security model for ACP GRASP from other ACP traffic. Instead, there is just another layer of protection against certain attacks from the inside which is important due to the role of GRASP in the ACP.

6.9. Context Separation

The ACP is in a separate context from the normal Data-Plane of the node. This context includes the ACP channels’ IPv6 forwarding and routing as well as any required higher layer ACP functions.

In classical network system, a dedicated so called Virtual routing and forwarding instance (VRF) is one logical implementation option for the ACP. If possible by the systems software architecture, separation options that minimize shared components are preferred, such as a logical container or virtual machine instance. The context for the ACP needs to be established automatically during bootstrap of a node. As much as possible it should be protected from being modified unintentionally by ("Data-Plane") configuration.

Context separation improves security, because the ACP is not reachable from the Data-Plane routing or forwarding table(s). Also, configuration errors from the Data-Plane setup do not affect the ACP.
6.10. Addressing inside the ACP

The channels explained above typically only establish communication between two adjacent nodes. In order for communication to happen across multiple hops, the autonomic control plane requires ACP network wide valid addresses and routing. Each ACP node must create a Loopback interface with an ACP network wide unique address inside the ACP context (as explained in in Section 6.9). This address may be used also in other virtual contexts.

With the algorithm introduced here, all ACP nodes in the same routing subdomain have the same /48 ULA prefix. Conversely, ULA global IDs from different domains are unlikely to clash, such that two ACP networks can be merged, as long as the policy allows that merge. See also Section 9.1 for a discussion on merging domains.

Links inside the ACP only use link-local IPv6 addressing, such that each nodes ACP only requires one routable virtual address.

6.10.1. Fundamental Concepts of Autonomic Addressing

- Usage: Autonomic addresses are exclusively used for self-management functions inside a trusted domain. They are not used for user traffic. Communications with entities outside the trusted domain use another address space, for example normally managed routable address space (called "Data-Plane" in this document).

- Separation: Autonomic address space is used separately from user address space and other address realms. This supports the robustness requirement.

- Loopback-only: Only ACP Loopback interfaces (and potentially those configured for "ACP connect", see Section 8.1) carry routable address(es); all other interfaces (called ACP virtual interfaces) only use IPv6 link local addresses. The usage of IPv6 link local addressing is discussed in [RFC7404].

- Use-ULA: For Loopback interfaces of ACP nodes, we use Unique Local Addresses (ULA), as defined in [RFC4193] with L=1 (as defined in section 3.1 of [RFC4193]). Note that the random hash for ACP Loopback addresses uses the definition in Section 6.10.2 and not the one of [RFC4193] section 3.2.2.

- No external connectivity: They do not provide access to the Internet. If a node requires further reaching connectivity, it should use another, traditionally managed address scheme in parallel.
Addresses in the ACP are permanent, and do not support temporary addresses as defined in [RFC4941].

Addresses in the ACP are not considered sensitive on privacy grounds because ACP nodes are not expected to be end-user host. All ACP nodes are in one (potentially federated) administrative domain. They are assumed to be to be candidate hosts of ACP traffic amongst each other or transit thereof. There are no transit nodes less privileged to know about the identity of other hosts in the ACP. Therefore, ACP addresses do not need to be pseudo-random as discussed in [RFC7721]. Because they are not propagated to untrusted (non ACP) nodes and stay within a domain (of trust), we also consider them not to be subject to scanning attacks.

The ACP is based exclusively on IPv6 addressing, for a variety of reasons:

- Simplicity, reliability and scale: If other network layer protocols were supported, each would have to have its own set of security associations, routing table and process, etc.

- Autonomic functions do not require IPv4: Autonomic functions and autonomic service agents are new concepts. They can be exclusively built on IPv6 from day one. There is no need for backward compatibility.

- OAM protocols do not require IPv4: The ACP may carry OAM protocols. All relevant protocols (SNMP, TFTP, SSH, SCP, Radius, Diameter, ...) are available in IPv6. See also [RFC8368] for how ACP could be made to interoperate with IPv4 only OAM.

6.10.2. The ACP Addressing Base Scheme

The Base ULA addressing scheme for ACP nodes has the following format:

```
+----+-------------------------+------+------------------------------+
|fd  | hash(routing-subdomain) | Type |     (sub-scheme)             |
+----+-------------------------+------+------------------------------+
```

**Figure 9: ACP Addressing Base Scheme**

The first 48-bits follow the ULA scheme, as defined in [RFC4193], to which a type field is added:

- "fd" identifies a locally defined ULA address.
The 40-bits ULA "global ID" (term from [RFC4193]) for ACP addresses carried in the domain information field of domain certificates are the first 40-bits of the SHA256 hash of the routing subdomain from the same domain information field. In the example of Section 6.1.1, the routing subdomain is "area51.research.acp.example.com" and the 40-bits ULA "global ID" 89b714f3db.

When creating a new routing-subdomain for an existing autonomic network, it MUST be ensured, that rsub is selected so the resulting hash of the routing-subdomain does not collide with the hash of any pre-existing routing-subdomains of the autonomic network. This ensures that ACP addresses created by registrars for different routing subdomains do not collide with each others.

To allow for extensibility, the fact that the ULA "global ID" is a hash of the routing subdomain SHOULD NOT be assumed by any ACP node during normal operations. The hash function is only executed during the creation of the certificate. If BRSKI is used then the BRSKI registrar will create the domain information field in response to the EST Certificate Signing Request (CSR) Attribute Request message by the pledge.

Establishing connectivity between different ACP (different acp-domain-name) is outside the scope of this specification. If it is being done through future extensions, then the rsub of all routing-subdomains across those autonomic networks need to be selected so their hashes do not collide. For example a large cooperation with its own private Trust Anchor may want to create different autonomic networks that initially should not be able to connect but where the option to do so should be kept open. When taking this future possibility into account, it is easy to always select rsub so that no collisions happen.

Type: This field allows different address sub-schemes. This addresses the "upgradability" requirement. Assignment of types for this field will be maintained by IANA.

The sub-scheme may imply a range or set of addresses assigned to the node, this is called the ACP address range/set and explained in each sub-scheme.

Please refer to Section 6.10.7 and Appendix A.1 for further explanations why the following Sub-Addressing schemes are used and why multiple are necessary.
6.10.3. ACP Zone Addressing Sub-Scheme

The sub-scheme defined here is defined by the Type value 00b (zero) in the base scheme and 0 in the Z bit.

```
+-----------------+---+---------++-----------------------------+---+
|  (base scheme)  | Z | Zone-ID ||           Node-ID               |
|                 |   |         || Registrar-ID |   Node-Number| V |
+-----------------+---+---------++--------------+--------------+---+
```

Figure 10: ACP Zone Addressing Sub-Scheme

The fields are defined as follows:

- **Zone-ID**: If set to all zero bits: The Node-ID bits are used as an identifier (as opposed to a locator). This results in a non-hierarchical, flat addressing scheme. Any other value indicates a zone. See Section 6.10.3.1 on how this field is used in detail.
- **Z**: MUST be 0.
- **Node-ID**: A unique value for each node.

The 64-bit Node-ID is derived and composed as follows:

- **Registrar-ID (48-bit)**: A number unique inside the domain that identifies the ACP registrar which assigned the Node-ID to the node. A MAC address of the ACP registrar can be used for this purpose.
- **Node-Number**: A number which is unique for a given ACP registrar, to identify the node. This can be a sequentially assigned number.
- **V (1-bit)**: Virtualization bit: 0: Indicates the ACP itself ("ACP node base system"); 1: Indicates the optional "host" context on the ACP node (see below).

In the ACP Zone Addressing Sub-Scheme, the ACP address in the certificate has Zone-ID and V fields as all zero bits. The ACP address set includes addresses with any Zone-ID value and any V value.

The "Node-ID" itself is unique in a domain (i.e., the Zone-ID is not required for uniqueness). Therefore, a node can be addressed either
as part of a flat hierarchy (Zone-ID = 0), or with an aggregation scheme (any other Zone-ID). An address with Zone-ID = 0 is an identifier, with a Zone-ID !=0 it is a locator. See Section 6.10.3.1 for more details.

The Virtual bit in this sub-scheme allows the easy addition of the ACP as a component to existing systems without causing problems in the port number space between the services in the ACP and the existing system. V:0 is the ACP router (autonomic node base system), V:1 is the host with pre-existing transport endpoints on it that could collide with the transport endpoints used by the ACP router. The ACP host could for example have a p2p virtual interface with the V:0 address as its router into the ACP. Depending on the software design of ASAs, which is outside the scope of this specification, they may use the V:0 or V:1 address.

The location of the V bit(s) at the end of the address allows the announcement of a single prefix for each ACP node. For example, in a network with 20,000 ACP nodes, this avoid 20,000 additional routes in the routing table.

6.10.3.1. Usage of the Zone-ID Field

The Zone-ID allows for the introduction of route prefixes in the addressing scheme.

Zone-ID = 0 is the default addressing scheme in an ACP domain. Every ACP node with a Zone Addressing Sub-Scheme address MUST respond to its ACP address with Zone-ID = 0. Used on its own this leads to a non-hierarchical address scheme, which is suitable for networks up to a certain size. Zone-ID = 0 addresses act as identifiers for the nodes, and aggregation of these address in the ACP routing table is not possible.

If aggregation is required, the 13-bit Zone-ID value allows for up to 8191 zones. The allocation of Zone-ID’s may either happen automatically through a to-be-defined algorithm; or it could be configured and maintained explicitly.

If a node learns (see Appendix A.10.1) that it is part of a zone, it MUST also respond to its ACP address with that Zone-ID. In this case the ACP Loopback is configured with two ACP addresses: One for Zone-ID = 0 and one for the assigned Zone-ID. This method allows for a smooth transition between a flat addressing scheme and a hierarchical one.

A node knowing it is in a zone MUST use that Zone-ID != 0 address in GRASP locator fields. This eliminates the use of the identifier
address (Zone-ID = 0) in forwarding and the need for network wide reachability of those non-aggregable identifier addresses. Zone-ID != 0 addresses are assumed to be aggregable in routing/forwarding based on how they are allocated in the ACP topology.

Note: The Zone-ID is one method to introduce structure or hierarchy into the ACP. Another way is the use of the routing subdomain field in the ACP that leads to multiple /48 Global IDs within an ACP domain.

Note: Zones and Zone-ID as defined here are not related to [RFC4007] zones or zone_id. ACP zone addresses are not scoped (reachable only from within an RFC4007 zone) but reachable across the whole ACP. An RFC4007 zone_id is a zone index that has only local significance on a node, whereas an ACP Zone-ID is an identifier for an ACP zone that is unique across that ACP.

6.10.4. ACP Manual Addressing Sub-Scheme

The sub-scheme defined here is defined by the Type value 00b (zero) in the base scheme and 1 in the Z bit.

```
+---------------------+---+----------++-----------------------------+
|    (base scheme)    | Z | Subnet-ID||     Interface Identifier    |
+---------------------+---+----------++-----------------------------+
```

Figure 11: ACP Manual Addressing Sub-Scheme

The fields are defined as follows:

- Subnet-ID: Configured subnet identifier.
- Z: MUST be 1.
- Interface Identifier.

This sub-scheme is meant for "manual" allocation to subnets where the other addressing schemes cannot be used. The primary use case is for assignment to ACP connect subnets (see Section 8.1.1).

"Manual" means that allocations of the Subnet-ID need to be done today with pre-existing, non-autonomic mechanisms. Every subnet that
uses this addressing sub-scheme needs to use a unique Subnet-ID (unless some anycast setup is done).

The Z bit field was added to distinguish Zone addressing and manual addressing sub-schemes without requiring one more bit in the base scheme and therefore allowing for the Vlong scheme (described below) to have one more bit available.

Manual addressing sub-scheme addresses SHOULD NOT be used in ACP domain certificates. Any node capable to build ACP secure channels and permitted by Registrar policy to participate in building ACP secure channels SHOULD receive an ACP address (prefix) from one of the other ACP addressing sub-schemes. Nodes not capable (or permitted) to participate in ACP secure channels can connect to the ACP via ACP connect interfaces of ACP edge nodes (see Section 8.1), without setting up an ACP secure channel. Their ACP domain certificate MUST include an empty acp-address to indicate that their ACP domain certificate is only usable for non-ACP secure channel authentication, such as end-to-end transport connections across the ACP or Data-Plane.

Address management of ACP connect subnets is done using traditional assignment methods and existing IPv6 protocols. See Section 8.1.3 for details.

6.10.5. ACP Vlong Addressing Sub-Scheme

The sub-scheme defined here is defined by the Type value 01b (one) in the base scheme.

```
+---------------------++-----------------------------+----------+
|    (base scheme)    ||           Node-ID                      |
|                     || Registrar-ID |   Node-Number|        V |
+---------------------++--------------+--------------+----------+
  50                46             24/16          8/16
```

Figure 12: ACP Vlong Addressing Sub-Scheme

This addressing scheme foregoes the Zone-ID field to allow for larger, flatter routed networks (e.g., as in IoT) with 8421376 Node-Numbers ($2^{23}+2^{15}$). It also allows for up to $2^{16}$ (i.e. 65536) different virtualized addresses within a node, which could be used to address individual software components in an ACP node.

The fields are the same as in the Zone-ID sub-scheme with the following refinements:
V: Virtualization field: 8 or 16 bit. Values 0 and 1 are assigned in the same way as in the Zone-ID sub-scheme, the other values are for further use by the node.

Registrar-ID: To maximize Node-Number and V, the Registrar-ID is reduced to 46-bits. This still permits the use of the MAC address of an ACP registrar by removing the V and U bits from the 48-bits of a MAC address (those two bits are never unique, so they cannot be used to distinguish MAC addresses).

If the first bit of the "Node-Number" is "1", then the Node-Number is 16-bit long and the V field is 16-bit long. Otherwise the Node-Number is 24-bit long and the V field is 8-bit long.

"0" bit Node-Numbers are intended to be used for "general purpose" ACP nodes that would potentially have a limited number (< 256) of clients (ASA/Autonomic Functions or legacy services) of the ACP that require separate V(irtual) addresses. "1" bit Node-Numbers are intended for ACP nodes that are ACP edge nodes (see Section 8.1.1) or that have a large number of clients requiring separate V(irtual) addresses. For example, large SDN controllers with container modular architecture (see Section 8.1.2).

In the Vlong addressing sub-scheme, the ACP address in the certificate has all V field bits as zero. The ACP address set for the node includes any V value.

6.10.6. Other ACP Addressing Sub-Schemes

Before further addressing sub-schemes are defined, experience with the schemes defined here should be collected. The schemes defined in this document have been devised to allow hopefully sufficiently flexible setup of ACPs for a variety of situation. These reasons also lead to the fairly liberal use of address space: The Zone Addressing Sub-Scheme is intended to enable optimized routing in large networks by reserving bits for Zone-ID’s. The Vlong addressing sub-scheme enables the allocation of 8/16-bit of addresses inside individual ACP nodes. Both address spaces allow distributed, uncoordinated allocation of node addresses by reserving bits for the registrar-ID field in the address.

IANA is asked need to assign a new "type" for each new addressing sub-scheme. With the current allocations, only 2 more schemes are possible, so the last addressing scheme MUST provide further extensions (e.g., by reserving bits from it for further extensions).
6.10.7. ACP Registrars

ACP registrars are responsible to enroll candidate ACP nodes with ACP domain certificates and associated trust point(s). They are also responsible that an ACP domain information field is included in the ACP domain certificate carrying the ACP domain name and the ACP nodes ACP address prefix. This address prefix is intended to persist unchanged through the lifetime of the ACP node.

Because of the ACP addressing sub-schemes, an ACP domain can have multiple distributed ACP registrars that do not need to coordinate for address assignment. ACP registrars can also be sub-CAs, in which case they can also assign ACP domain certificates without dependencies against a (shared) root-CA (except during renewals of their own certificates).

ACP registrars are PKI registration authorities (RA) enhanced with the handling of the ACP domain certificate specific fields. They request certificates for ACP nodes from a Certificate Authority through any appropriate mechanism (out of scope in this document, but required to be BRSKI for ANI registrars). Only nodes that are trusted to be compliant with the requirements against registrar described in this section must be given the necessary credentials to perform this RA function, such as credentials for the BRSKI connection to the CA for ANI registrars.

6.10.7.1. Use of BRSKI or other Mechanism/Protocols

Any protocols or mechanisms may be used as ACP registrars, as long as the resulting ACP certificate and trust anchors allow to perform the ACP domain membership described in Section 6.1.2 with other ACP domain members, and meet the ACP addressing requirements for its ACP domain information field as described further below in this section.

An ACP registrar could be a person deciding whether to enroll a candidate ACP node and then orchestrating the enrollment of the ACP certificate and associated trust anchor, using command line or web based commands on the candidate ACP node and trust anchor to generate and sign the ACP domain certificate and configure certificate and trust anchors onto the node.

The only currently defined protocol for ACP registrars is BRSKI ([I-D.ietf-anima-bootstrapping-keyinfra]). When BRSKI is used, the ACP nodes are called ANI nodes, and the ACP registrars are called BRSKI or ANI registrars. The BRSKI specification does not define the handling of the ACP domain information field because the rules do not depend on BRSKI but apply equally to any protocols/mechanisms an ACP registrar may use.
6.10.7.2. Unique Address/Prefix allocation

ACP registrars MUST NOT allocate ACP address prefixes to ACP nodes via the ACP domain information field that would collide with the ACP address prefixes of other ACP nodes in the same ACP domain. This includes both prefixes allocated by the same ACP registrar to different ACP nodes as well as prefixes allocated by other ACP registrars for the same ACP domain.

For this purpose, an ACP registrar MUST have one or more unique 46-bit identifiers called Registrar-IDs used to allocate ACP address prefixes. The lower 46-bits of a EUI-48 MAC addresses are globally unique 46 bit identifiers, so ACP registrars with known unique EUI-48 MAC addresses can use these as Registrar-IDs. Registrar-IDs do not need to be globally unique but only unique across the set of ACP registrars for an ACP domain, so other means to assign unique Registrar-IDs to ACP registrars can be used, such as configuration on the ACP registrars.

When the candidate ACP device (called Pledge in BRSKI) is to be enrolled into an ACP domain, the ACP registrar needs to allocate a unique ACP address to the node and ensure that the ACP certificate gets a domain information field (Section 6.1.1) with the appropriate information - ACP domain-name, ACP-address, and so on. If the ACP registrar uses BRSKI, it signals the ACP domain information field to the Pledge via the EST /csraddrs command (see [I-D.ietf-anima-bootstrapping-keyinfra], section 5.8.2 - "EST CSR Attributes").

[RFC Editor: please update reference to section 5.8.2 accordingly with latest BRSKI draft at time of publishing, or RFC]

6.10.7.3. Addressing Sub-Scheme Policies

The ACP registrar selects for the candidate ACP node a unique address prefix from an appropriate ACP addressing sub-scheme, either a zone addressing sub-scheme prefix (see Section 6.10.3), or a Vlong addressing sub-scheme prefix (see Section 6.10.5). The assigned ACP address prefix encoded in the domain information field of the ACP domain certificate indicates to the ACP node its ACP address information. The sub-addressing scheme indicates the prefix length: /127 for zone address sub-scheme, /120 or /112 for Vlong address sub-scheme. The first address of the prefix is the ACP address, all other addresses in the prefix are for other uses by the ACP node as described in the zone and Vlong addressing sub-scheme sections. The ACP address prefix itself is then signaled by the ACP node into the ACP routing protocol (see Section 6.11) to establish IPv6 reachability across the ACP.
The choice of addressing sub-scheme and prefix-length in the Vlong address sub-scheme is subject to ACP registrar policy. It could be an ACP domain wide policy, or a per ACP node or per ACP node type policy. For example, in BRSKI, the ACP registrar is aware of the IDevID of the candidate ACP node, which contains a serialNnumber that is typically indicating the nodes vendor and device type and can be used to drive a policy selecting an appropriate addressing sub-scheme for the (class of) node(s).

ACP registrars SHOULD default to allocate ACP zone sub-address scheme addresses with Subnet-ID 0. Allocation and use of zone sub-addresses with Subnet-ID != 0 is outside the scope of this specification because it would need to go along with rules for extending ACP routing to multiple zones, which is outside the scope of this specification.

ACP registrars that can use the IDevID of a candidate ACP device SHOULD be able to choose the zone vs. Vlong sub-address scheme for ACP nodes based on the serialNumber of the IDevID, for example by the PID (Product Identifier) part which identifies the product type, or the complete serialNumber.

In a simple allocation scheme, an ACP registrar remembers persistently across reboots its currently used Registrar-ID and for each addressing scheme (zone with Subnet-ID 0, Vlong with /112, Vlong with /120), the next Node-Number available for allocation and increases it during successful enrollment to an ACP node. In this simple allocation scheme, the ACP registrar would not recycle ACP address prefixes from no longer used ACP nodes.

6.10.7.4. Address/Prefix Persistence

When an ACP domain certificate is renewed or rekeyed via EST or other mechanisms, the ACP address/prefix in the ACP domain information field MUST be maintained unless security issues or violations of the unique address assignment requirements exist or are suspected by the ACP registrar.

ACP address information SHOULD be maintained even when the renewing/rekeying ACP registrar is not the same as the one that enrolled the prior ACP certificate. See Section 10.2.4 for an example.

ACP address information SHOULD also be maintained even after an ACP certificate did expire or failed. See Section 6.1.4.5 and Section 6.1.4.6.
6.10.7.5. Further Details

Section 10.2 discusses further informative details of ACP registrars: what interactions registrars need, what parameters they require, certificate renewal and limitations, use of sub-CAs on registrars and centralized policy control.

6.11. Routing in the ACP

Once ULA address are set up all autonomic entities should run a routing protocol within the autonomic control plane context. This routing protocol distributes the ULA created in the previous section for reachability. The use of the autonomic control plane specific context eliminates the probable clash with Data-Plane routing tables and also secures the ACP from interference from the configuration mismatch or incorrect routing updates.

The establishment of the routing plane and its parameters are automatic and strictly within the confines of the autonomic control plane. Therefore, no explicit configuration is required.

All routing updates are automatically secured in transit as the channels of the ACP are encrypted, and this routing runs only inside the ACP.

The routing protocol inside the ACP is RPL ([RFC6550]). See Appendix A.4 for more details on the choice of RPL.

RPL adjacencies are set up across all ACP channels in the same domain including all its routing subdomains. See Appendix A.7 for more details.

6.11.1. RPL Profile

The following is a description of the RPL profile that ACP nodes need to support by default. The format of this section is derived from draft-ietf-roll-applicability-template.

6.11.1.1. Overview

The chosen RPL profile is one that expects a fairly reliable network with reasonably fast links so that RPL convergence will be triggered immediately upon recognition of link failure/recovery.

The profile is also designed to not require any RPL Data-Plane artifacts (such as defined in [RFC6553]). This is largely driven by the desire to avoid introducing the required Hop-by-Hop headers into the ACP forwarding plane, especially to support devices with silicon
forwarding planes that cannot support insertion/removal of these
headers in silicon or hop-by-hop forwarding based on them. Note:
Insertion/removal of headers by a (potentially silicon based) ACP
node would be be necessary when senders/receivers of ACP packets are
legacy NOC devices connected via ACP connect (see Section 8.1.1 to
the ACP. Their connectivity can be handled in RPL as non-RPL-aware
leafs (or "Internet") according to the Data-Plane architecture
explained in [I-D.ietf-roll-useofrplinfo].

To avoid Data-Plane artefacts, the profile uses a simple destination
prefix based routing/forwarding table. To achieve this, the profiles
uses only one RPL instanceID. This single instanceID can contain
only one Destination Oriented Directed Acyclic Graph (DODAG), and the
routing/forwarding table can therefore only calculate a single class
of service ("best effort towards the primary NOC/root") and cannot
create optimized routing paths to accomplish latency or energy goals
between any two nodes.

Consider a network that has multiple NOCs in different locations.
Only one NOC will become the DODAG root. Traffic to and from other
NOCs has to be sent through the DODAG (shortest path tree) rooted in
the primary NOC. Depending on topology, this can be an annoyance
from a latency point of view or from minimizing network path
resources, but this is deemed to be acceptable given how ACP traffic
is "only" network management/control traffic.

Using a single instanceID/DODAG does not introduce a single point of
failure, as the DODAG will reconfigure itself when it detects data-
plane forwarding failures including choosing a different root when
the primary one fails. See Appendix A.10.4 for more details.

The benefit of this profile, especially compared to other IGP is
that it does not calculate routes for node reachable through the same
interface as the DODAG root. This RPL profile can therefore scale to
much larger number of ACP nodes in the same amount of compute and
memory than other routing protocols. Especially on nodes that are
leafs of the topology or those close to those leafs.

The lack of RPL Packet Information (RPI, the IPv6 header for RPL
defined by [RFC6553]), means that the Data-Plane will have no rank
value that can be used to detect loops. As a result, traffic may
loop until the time-to-live (TTL) of the packet reaches zero. This
is the same behavior as that of other IGP that do not have the Data-
Plane options of RPL.

Since links in the ACP are assumed to be mostly reliable (or have
link layer protection against loss) and because there is no stretch
according to Section 6.11.1.7, loops caused by RPL routing packet loss should be exceedingly rare.

There are a variety of mechanisms possible in RPL to further avoid temporary loops: DODAG Information Objects (DIOs) SHOULD be sent 2...3 times to inform children when losing the last parent. The technique in [RFC6550] section 8.2.2.6. (Detaching) SHOULD be favored over that in section 8.2.2.5., (Poisoning) because it allows local connectivity. Nodes SHOULD select more than one parent, at least 3 if possible, and send Destination Advertisement Objects (DAO)s to all of them in parallel.

Additionally, failed ACP tunnels can be quickly discovered the secure channel protocol mechanisms such as IKEv2 Dead Peer Detection. This can function as a replacement for a Low-power and Lossy Networks’ (LLN’s) Expected Transmission Count (ETX) feature that is not used in this profile. A failure of an ACP tunnel should immediately signal the RPL control plane to pick a different parent.

6.11.1.2. RPL Instances

Single RPL instance. Default RPLInstanceID = 0.

6.11.1.3. Storing vs. Non-Storing Mode

RPL Mode of Operations (MOP): MUST support mode 2 - "Storing Mode of Operations with no multicast support". Implementations MAY support mode 3 ("... with multicast support" as that is a superset of mode 2). Note: Root indicates mode in DIO flow.

6.11.1.4. DAO Policy

Proactive, aggressive DAO state maintenance:

- Use K-flag in unsolicited DAO indicating change from previous information (to require DAO-ACK).
- Retry such DAO DAO-RETRIES(3) times with DAO-ACK_TIME_OUT(256ms) in between.

6.11.1.5. Path Metric

Hopcount.
6.11.1.6. Objective Function

Objective Function (OF): Use OF0 [RFC6552]. No use of metric containers.

rank_factor: Derived from link speed: <= 100Mbps: LOW_SPEED_FACTOR(5), else HIGH_SPEED_FACTOR(1)

6.11.1.7. DODAG Repair

Global Repair: we assume stable links and ranks (metrics), so no need to periodically rebuild DODAG. DODAG version only incremented under catastrophic events (e.g., administrative action).

Local Repair: As soon as link breakage is detected, send No-Path DAO for all the targets that were reachable only via this link. As soon as link repair is detected, validate if this link provides you a better parent. If so, compute your new rank, and send new DIO that advertises your new rank. Then send a DAO with a new path sequence about yourself.

stretch_rank: none provided ("not stretched").

Data Path Validation: Not used.

Trickle: Not used.

6.11.1.8. Multicast

Not used yet but possible because of the selected mode of operations.

6.11.1.9. Security

[RFC6550] security not used, substituted by ACP security.

Because the ACP links already include provisions for confidentiality and integrity protection, their usage at the RPL layer would be redundant, and so RPL security is not used.

6.11.1.10. P2P communications

Not used.

6.11.1.11. IPv6 address configuration

Every ACP node (RPL node) announces an IPv6 prefix covering the address(es) used in the ACP node. The prefix length depends on the chosen addressing sub-scheme of the ACP address provisioned into the
certificate of the ACP node, e.g., /127 for Zone Addressing Sub-Scheme or /112 or /120 for Vlong addressing sub-scheme. See Section 6.10 for more details.

Every ACP node MUST install a black hole (aka null) route for whatever ACP address space that it advertises (i.e.: the /96 or /127). This is avoid routing loops for addresses that an ACP node has not (yet) used.

6.11.1.12. Administrative parameters

Administrative Preference ([RFC6550], 3.2.6 - to become root): Indicated in DoDAGPreference field of DIO message.

- Explicit configured "root": 0b100
- ACP registrar (Default): 0b011
- ACP-connect (non-registrar): 0b010
- Default: 0b001.

6.11.1.13. RPL Data-Plane artifacts

RPI (RPL Packet Information [RFC6553]): Not used as there is only a single instance, and data path validation is not being used.

SRH (RPL Source Routing - RFC6552): Not used. Storing mode is being used.

6.11.1.14. Unknown Destinations

Because RPL minimizes the size of the routing and forwarding table, prefixes reachable through the same interface as the RPL root are not known on every ACP node. Therefore traffic to unknown destination addresses can only be discovered at the RPL root. The RPL root SHOULD have attach safe mechanisms to operationally discover and log such packets.

6.12. General ACP Considerations

Since channels are by default established between adjacent neighbors, the resulting overlay network does hop-by-hop encryption. Each node decrypts incoming traffic from the ACP, and encrypts outgoing traffic to its neighbors in the ACP. Routing is discussed in Section 6.11.
6.12.1. Performance

There are no performance requirements against ACP implementations defined in this document because the performance requirements depend on the intended use case. It is expected that full autonomic node with a wide range of ASA can require high forwarding plane performance in the ACP, for example for telemetry. Implementations of ACP to solely support traditional/SDN style use cases can benefit from ACP at lower performance, especially if the ACP is used only for critical operations, e.g., when the Data-Plane is not available. The design of the ACP as specified in this document is intended to support a wide range of performance options: It is intended to allow software-only implementations at potentially low performance, but can also support high performance options. See [RFC8368] for more details.

6.12.2. Addressing of Secure Channels

In order to be independent of the Data-Plane (routing and addressing) the GRASP discovered (autonomic) ACP secure channels use IPv6 link local addresses between adjacent neighbors. Note: Section 8.2 specifies extensions in which secure channels are configured tunnels operating over the Data-Plane, so those secure channels cannot be independent of the Data-Plane.

To avoid that Data-Plane configuration can impact the operations of the IPv6 (link-local) interface/address used for ACP channels, appropriate implementation considerations are required. If the IPv6 interface/link-local address is shared with the Data-Plane it needs to be impossible to unconfigure/disable it through configuration. Instead of sharing the IPv6 interface/link-local address, a separate (virtual) interface with a separate IPv6 link-local address can be used. For example, the ACP interface could be run over a separate MAC address of an underlying L2 (Ethernet) interface. For more details and options, see Appendix A.10.2.

Note that other (non-ideal) implementation choices may introduce additional undesired dependencies against the Data-Plane. For example shared code and configuration of the secure channel protocols (IPsec / DTLS).

6.12.3. MTU

The MTU for ACP secure channels must be derived locally from the underlying link MTU minus the secure channel encapsulation overhead.

ACP secure Channel protocols do not need to perform MTU discovery because they are built across L2 adjacencies - the MTU on both sides...
connecting to the L2 connection are assumed to be consistent. Extensions to ACP where the ACP is for example tunneled need to consider how to guarantee MTU consistency. This is an issue of tunnels, not an issue of running the ACP across a tunnel. Transport stacks running across ACP can perform normal PMTUD (Path MTU Discovery). Because the ACP is meant to be prioritize reliability over performance, they MAY opt to only expect IPv6 minimum MTU (1280) to avoid running into PMTUD implementation bugs or underlying link MTU mismatch problems.

6.12.4. Multiple links between nodes

If two nodes are connected via several links, the ACP SHOULD be established across every link, but it is possible to establish the ACP only on a sub-set of links. Having an ACP channel on every link has a number of advantages, for example it allows for a faster failover in case of link failure, and it reflects the physical topology more closely. Using a subset of links (for example, a single link), reduces resource consumption on the node, because state needs to be kept per ACP channel. The negotiation scheme explained in Section 6.5 allows Alice (the node with the higher ACP address) to drop all but the desired ACP channels to Bob - and Bob will not retry to build these secure channels from his side unless Alice shows up with a previously unknown GRASP announcement (e.g., on a different link or with a different address announced in GRASP).

6.12.5. ACP interfaces

The ACP VRF has conceptually two type of interfaces: The "ACP Loopback interface(s)" to which the ACP ULA address(es) are assigned and the "ACP virtual interfaces" that are mapped to the ACP secure channels.

The term "Loopback interface" was introduced initially to refer to an internal interface on a node that would allow IP traffic between transport endpoints on the node in the absence or failure of any or all external interfaces, see [RFC4291] section 2.5.3.

Even though Loopback interfaces were originally designed to hold only Loopback addresses not reachable from outside the node, these interfaces are also commonly used today to hold addresses reachable from the outside. They are meant to be reachable independent of any external interface being operational, and therefore to be more resilient. These addresses on Loopback interfaces can be thought of as "node addresses" instead of "interface addresses", and that is what ACP address(es) are. This construct makes it therefore possible to address ACP nodes with a well-defined set of addresses independent of the number of external interfaces.
For these reason, the ACP (ULA) address(es) are assigned to Loopback interface(s).

Any type of ACP secure channels to another ACP node can be mapped to ACP virtual interfaces in following ways. This is independent of the chosen secure channel protocol (IPsec, DTLS or other future protocol - standards or non-standards):

ACP point-to-point virtual interface:

Each ACP secure channel is mapped into a separate point-to-point ACP virtual interface. If a physical subnet has more than two ACP capable nodes (in the same domain), this implementation approach will lead to a full mesh of ACP virtual interfaces between them.

ACP multi-access virtual interface:

In a more advanced implementation approach, the ACP will construct a single multi-access ACP virtual interface for all ACP secure channels to ACP capable nodes reachable across the same underlying (physical) subnet. IPv6 link-local multicast packets sent into an ACP multi-access virtual interface are replicated to every ACP secure channel mapped into the ACP multicast-access virtual interface. IPv6 unicast packets sent into an ACP multi-access virtual interface are sent to the ACP secure channel that belongs to the ACP neighbor that is the next-hop in the ACP forwarding table entry used to reach the packets destination address.

There is no requirement for all ACP nodes on the same multi-access subnet to use the same type of ACP virtual interface. This is purely a node local decision.

ACP nodes MUST perform standard IPv6 operations across ACP virtual interfaces including SLAAC (Stateless Address Auto-Configuration) - [RFC4862]) to assign their IPv6 link local address on the ACP virtual interface and ND (Neighbor Discovery - [RFC4861]) to discover which IPv6 link-local neighbor address belongs to which ACP secure channel mapped to the ACP virtual interface. This is independent of whether the ACP virtual interface is point-to-point or multi-access.

"Optimistic Duplicate Address Detection (DAD)" according to [RFC4429] is RECOMMENDED because the likelihood for duplicates between ACP nodes is highly improbable as long as the address can be formed from a globally unique local assigned identifier (e.g., EUI-48/EUI-64, see below).

ACP nodes MAY reduce the amount of link-local IPv6 multicast packets from ND by learning the IPv6 link-local neighbor address to ACP
secure channel mapping from other messages such as the source address of IPv6 link-local multicast RPL messages - and therefore forego the need to send Neighbor Solicitation messages.

The ACP virtual interface IPv6 link local address can be derived from any appropriate local mechanism such as node local EUI-48 or EUI-64 ("EUI" stands for "Extended Unique Identifier"). It MUST NOT depend on something that is attackable from the Data-Plane such as the IPv6 link-local address of the underlying physical interface, which can be attacked by SLAAC, or parameters of the secure channel encapsulation header that may not be protected by the secure channel mechanism.

The link-layer address of an ACP virtual interface is the address used for the underlying interface across which the secure tunnels are built, typically Ethernet addresses. Because unicast IPv6 packets sent to an ACP virtual interface are not sent to a link-layer destination address but rather an ACP secure channel, the link-layer address fields SHOULD be ignored on reception and instead the ACP secure channel from which the message was received should be remembered.

Multi-access ACP virtual interfaces are preferable implementations when the underlying interface is a (broadcast) multi-access subnet because they do reflect the presence of the underlying multi-access subnet into the virtual interfaces of the ACP. This makes it for example simpler to build services with topology awareness inside the ACP VRF in the same way as they could have been built running natively on the multi-access interfaces.

Consider also the impact of point-to-point vs. multi-access virtual interface on the efficiency of flooding via link local multicasted messages:

Assume a LAN with three ACP neighbors, Alice, Bob and Carol. Alice’s ACP GRASP wants to send a link-local GRASP multicast message to Bob and Carol. If Alice’s ACP emulates the LAN as one point-to-point virtual interface to Bob and one to Carol, The sending application itself will send two copies, if Alice’s ACP emulates a LAN, GRASP will send one packet and the ACP will replicate it. The result is the same. The difference happens when Bob and Carol receive their packet. If they use ACP point-to-point virtual interfaces, their GRASP instance would forward the packet from Alice to each other as part of the GRASP flooding procedure. These packets are unnecessary and would be discarded by GRASP on receipt as duplicates (by use of the GRASP Session ID). If Bob and Carol’s ACP would emulate a multi-access virtual interface, then this would not happen, because GRASPs flooding procedure does not replicate back packets to the interface that they were received from.
Note that link-local GRASP multicast messages are not sent directly as IPv6 link-local multicast UDP messages into ACP virtual interfaces, but instead into ACP GRASP virtual interfaces, that are layered on top of ACP virtual interfaces to add TCP reliability to link-local multicast GRASP messages. Nevertheless, these ACP GRASP virtual interfaces perform the same replication of message and, therefore, result in the same impact on flooding. See Section 6.8.2 for more details.

RPL does support operations and correct routing table construction across non-broadcast multi-access (NBMA) subnets. This is common when using many radio technologies. When such NBMA subnets are used, they MUST NOT be represented as ACP multi-access virtual interfaces because the replication of IPv6 link-local multicast messages will not reach all NBMA subnet neighbors. In result, GRASP message flooding would fail. Instead, each ACP secure channel across such an interface MUST be represented as a ACP point-to-point virtual interface. See also Appendix A.10.4.

Care must also be taken when creating multi-access ACP virtual interfaces across ACP secure channels between ACP nodes in different domains or routing subdomains. The policies to be negotiated may be described as peer-to-peer policies in which case it is easier to create ACP point-to-point virtual interfaces for these secure channels.

7. ACP support on L2 switches/ports (Normative)

7.1. Why (Benefits of ACP on L2 switches)

Consider a large L2 LAN with ANrtr1...ANrtrN connected via some topology of L2 switches. Examples include large enterprise campus networks with an L2 core, IoT networks or broadband aggregation networks which often have even a multi-level L2 switched topology.

If the discovery protocol used for the ACP is operating at the subnet level, every ACP router will see all other ACP routers on the LAN as neighbors and a full mesh of ACP channels will be built. If some or all of the AN switches are autonomic with the same discovery protocol, then the full mesh would include those switches as well.
A full mesh of ACP connections can create fundamental scale challenges. The number of security associations of the secure channel protocols will likely not scale arbitrarily, especially when they leverage platform accelerated encryption/decryption. Likewise, any other ACP operations (such as routing) needs to scale to the number of direct ACP neighbors. An ACP router with just 4 physical interfaces might be deployed into a LAN with hundreds of neighbors connected via switches. Introducing such a new unpredictable scaling factor requirement makes it harder to support the ACP on arbitrary platforms and in arbitrary deployments.

Predictable scaling requirements for ACP neighbors can most easily be achieved if in topologies such as these, ACP capable L2 switches can ensure that discovery messages terminate on them so that neighboring ACP routers and switches will only find the physically connected ACP L2 switches as their candidate ACP neighbors. With such a discovery mechanism in place, the ACP and its security associations will only need to scale to the number of physical interfaces instead of a potentially much larger number of "LAN-connected" neighbors. And the ACP topology will follow directly the physical topology, something which can then also be leveraged in management operations or by ASAs.

In the example above, consider ANswitch1 and ANswitchM are ACP capable, and ANswitch2 is not ACP capable. The desired ACP topology is that ANrtr1 and ANrtrM only have an ACP connection to ANswitch1, and that ANswitch1, ANrtr2, ANrtrN have a full mesh of ACP connection amongst each other. ANswitch1 also has an ACP connection with ANswitchM and ANswitchM has ACP connections to anything else behind it.

7.2. How (per L2 port DULL GRASP)

To support ACP on L2 switches or L2 switched ports of an L3 device, it is necessary to make those L2 ports look like L3 interfaces for the ACP implementation. This primarily involves the creation of a separate DULL GRASP instance/domain on every such L2 port. Because GRASP has a dedicated link-local IPv6 multicast address (ALL_GRASP_NEIGHBORS), it is sufficient that all packets for this address are being extracted at the port level and passed to that DULL GRASP instance. Likewise the IPv6 link-local multicast packets sent by that DULL GRASP instance need to be sent only towards the L2 port for this DULL GRASP instance.

If the device with L2 ports is supporting per L2 port ACP DULL GRASP as well as MLD snooping ([RFC4541]), then MLD snooping must be changed to never forward packets for ALL_GRASP_NEIGHBORS because that would cause the problem that per L2 port ACP DULL GRASP is meant to overcome (forwarding DULL GRASP packets across L2 ports).
The rest of ACP operations can operate in the same way as in L3 devices: Assume for example that the device is an L3/L2 hybrid device where L3 interfaces are assigned to VLANs and each VLAN has potentially multiple ports. DULL GRASP is run as described individually on each L2 port. When it discovers a candidate ACP neighbor, it passes its IPv6 link-local address and supported secure channel protocols to the ACP secure channel negotiation that can be bound to the L3 (VLAN) interface. It will simply use link-local IPv6 multicast packets to the candidate ACP neighbor. Once a secure channel is established to such a neighbor, the virtual interface to which this secure channel is mapped should then actually be the L2 port and not the L3 interface to best map the actual physical topology into the ACP virtual interfaces. See Section 6.12.5 for more details about how to map secure channels into ACP virtual interfaces. Note that a single L2 port can still have multiple ACP neighbors if it connect for example to multiple ACP neighbors via a non-ACP enabled switch. The per L2 port ACP virtual interface can therefore still be a multi-access virtual LAN.

For example, in the above picture, ANswitch1 would run separate DULL GRASP instances on its ports to ANrtr1, ANswitch2 and ANswitchI, even though all those three ports may be in the data plane in the same (V)LAN and perform L2 switching between these ports, ANswitch1 would perform ACP L3 routing between them.

The description in the previous paragraph was specifically meant to illustrate that on hybrid L3/L2 devices that are common in enterprise, IoT and broadband aggregation, there is only the GRASP packet extraction (by Ethernet address) and GRASP link-local multicast per L2-port packet injection that has to consider L2 ports at the hardware forwarding level. The remaining operations are purely ACP control plane and setup of secure channels across the L3 interface. This hopefully makes support for per-L2 port ACP on those hybrid devices easy.

This L2/L3 optimized approach is subject to "address stealing", e.g., where a device on one port uses addresses of a device on another port. This is a generic issue in L2 LANs and switches often already have some form of "port security" to prohibit this. They rely on NDP or DHCP learning of which port/MAC-address and IPv6 address belong together and block duplicates. This type of function needs to be enabled to prohibit DoS attacks. Likewise the GRASP DULL instance needs to ensure that the IPv6 address in the locator-option matches the source IPv6 address of the DULL GRASP packet.

In devices without such a mix of L2 port/interfaces and L3 interfaces (to terminate any transport layer connections), implementation details will differ. Logically most simply every L2 port is
considered and used as a separate L3 subnet for all ACP operations. The fact that the ACP only requires IPv6 link-local unicast and multicast should make support for it on any type of L2 devices as simple as possible.

A generic issue with ACP in L2 switched networks is the interaction with the Spanning Tree Protocol. Without further L2 enhancements, the ACP would run only across the active STP topology and the ACP would be interrupted and re-converge with STP changes. Ideally, ACP peering should be built also across ports that are blocked in STP so that the ACP does not depend on STP and can continue to run unaffected across STP topology changes, where re-convergence can be quite slow. The above described simple implementation options are not sufficient to achieve this.

8. Support for Non-ACP Components (Normative)

8.1. ACP Connect

8.1.1. Non-ACP Controller / NMS system

The Autonomic Control Plane can be used by management systems, such as controllers or network management system (NMS) hosts (henceforth called simply "NMS hosts"), to connect to devices (or other type of nodes) through it. For this, an NMS host must have access to the ACP. The ACP is a self-protecting overlay network, which allows by default access only to trusted, autonomic systems. Therefore, a traditional, non-ACP NMS system does not have access to the ACP by default, such as any other external node.

If the NMS host is not autonomic, i.e., it does not support autonomic negotiation of the ACP, then it can be brought into the ACP by explicit configuration. To support connections to adjacent non-ACP nodes, an ACP node must support "ACP connect" (sometimes also called "autonomic connect"): "ACP connect" is an interface level configured workaround for connection of trusted non-ACP nodes to the ACP. The ACP node on which ACP connect is configured is called an "ACP edge node". With ACP connect, the ACP is accessible from those non-ACP nodes (such as NOC systems) on such an interface without those non-ACP nodes having to support any ACP discovery or ACP channel setup. This is also called "native" access to the ACP because to those (NOC) systems the interface looks like a normal network interface (without any encryption/novel-signaling).
ACP connect has security consequences: All systems and processes connected via ACP connect have access to all ACP nodes on the entire ACP, without further authentication. Thus, the ACP connect interface and (NOC) systems connected to it must be physically controlled/secured. For this reason the mechanisms described here do explicitly not include options to allow for a non-ACP router to be connected across an ACP connect interface and addresses behind such a router routed inside the ACP.

An ACP connect interface provides exclusively access to only the ACP. This is likely insufficient for many NMS hosts. Instead, they would require a second "Data-Plane" interface outside the ACP for connections between the NMS host and administrators, or Internet based services, or for direct access to the Data-Plane. The document "Using Autonomic Control Plane for Stable Connectivity of Network OAM" [RFC8368] explains in more detail how the ACP can be integrated in a mixed NOC environment.

An ACP connect interface SHOULD use an IPv6 address/prefix from the ACP Manual Addressing Sub-Scheme (Section 6.10.4), letting the operator configure for example only the Subnet-ID and having the node automatically assign the remaining part of the prefix/address. It SHOULD NOT use a prefix that is also routed outside the ACP so that the addresses clearly indicate whether it is used inside the ACP or not.
The prefix of ACP connect subnets MUST be distributed by the ACP edge node into the ACP routing protocol (RPL). The NMS hosts MUST connect to prefixes in the ACP routing table via its ACP connect interface. In the simple case where the ACP uses only one ULA prefix and all ACP connect subnets have prefixes covered by that ULA prefix, NMS hosts can rely on [RFC6724] to determine longest match prefix routes towards its different interfaces, ACP and data-plane. With RFC6724, the NMS host will select the ACP connect interface for all addresses in the ACP because any ACP destination address is longest matched by the address on the ACP connect interface. If the NMS hosts ACP connect interface uses another prefix or if the ACP uses multiple ULA prefixes, then the NMS hosts require (static) routes towards the ACP interface for these prefixes.

When an ACP Edge node receives a packet from an ACP connect interface, it MUST only forward it into the ACP if it has an IPv6 source address from that interface. This is sometimes called "RPF filtering". This MAY be changed through administrative measures.

To limit the security impact of ACP connect, nodes supporting it SHOULD implement a security mechanism to allow configuration/use of ACP connect interfaces only on nodes explicitly targeted to be deployed with it (those in physically secure locations such as a NOC). For example, the registrar could disable the ability to enable ACP connect on devices during enrollment and that property could only be changed through re-enrollment. See also Appendix A.10.5.

8.1.2. Software Components

The ACP connect mechanism be only be used to connect physically external systems (NMS hosts) to the ACP but also other applications, containers or virtual machines. In fact, one possible way to eliminate the security issue of the external ACP connect interface is to collocate an ACP edge node and an NMS host by making one a virtual machine or container inside the other; and therefore converting the unprotected external ACP subnet into an internal virtual subnet in a single device. This would ultimately result in a fully ACP enabled NMS host with minimum impact to the NMS hosts software architecture. This approach is not limited to NMS hosts but could equally be applied to devices consisting of one or more VNF (virtual network functions): An internal virtual subnet connecting out-of-band management interfaces of the VNFs to an ACP edge router VNF.

The core requirement is that the software components need to have a network stack that permits access to the ACP and optionally also the Data-Plane. Like in the physical setup for NMS hosts this can be realized via two internal virtual subnets. One that is connecting to
the ACP (which could be a container or virtual machine by itself), and one (or more) connecting into the Data-Plane.

This "internal" use of ACP connect approach should not considered to be a "workaround" because in this case it is possible to build a correct security model: It is not necessary to rely on unprovable external physical security mechanisms as in the case of external NMS hosts. Instead, the orchestration of the ACP, the virtual subnets and the software components can be done by trusted software that could be considered to be part of the ANI (or even an extended ACP). This software component is responsible for ensuring that only trusted software components will get access to that virtual subnet and that only even more trusted software components will get access to both the ACP virtual subnet and the Data-Plane (because those ACP users could leak traffic between ACP and Data-Plane). This trust could be established for example through cryptographic means such as signed software packages.

8.1.3. Auto Configuration

ACP edge nodes, NMS hosts and software components that as described in the previous section are meant to be composed via virtual interfaces SHOULD support on the ACP connect subnet StateLess Address Autoconfiguration (SLAAC - [RFC4862]) and route auto configuration according to [RFC4191].

The ACP edge node acts as the router on the ACP connect subnet, providing the (auto-)configured prefix for the ACP connect subnet to NMS hosts and/or software components. The ACP edge node uses route prefix option of RFC4191 to announce the default route (::/) with a lifetime of 0 and aggregated prefixes for routes in the ACP routing table with normal lifetimes. This will ensure that the ACP edge node does not become a default router, but that the NMS hosts and software components will route the prefixes used in the ACP to the ACP edge node.

Aggregated prefix means that the ACP edge node needs to only announce the /48 ULA prefixes used in the ACP but none of the actual /64 (Manual Addressing Sub-Scheme), /127 (ACP Zone Addressing Sub-Scheme), /112 or /120 (Vlong Addressing Sub-Scheme) routes of actual ACP nodes. If ACP interfaces are configured with non ULA prefixes, then those prefixes cannot be aggregated without further configured policy on the ACP edge node. This explains the above recommendation to use ACP ULA prefix covered prefixes for ACP connect interfaces: They allow for a shorter list of prefixes to be signaled via RFC4191 to NMS hosts and software components.
The ACP edge nodes that have a Vlong ACP address MAY allocate a subset of their /112 or /120 address prefix to ACP connect interface(s) to eliminate the need to non-autonomically configure/provision the address prefixes for such ACP connect interfaces.

8.1.4. Combined ACP/Data-Plane Interface (VRF Select)

Using two physical and/or virtual subnets (and therefore interfaces) into NMS Hosts (as per Section 8.1.1) or Software (as per Section 8.1.2) may be seen as additional complexity, for example with legacy NMS Hosts that support only one IP interface.

To provide a single subnet into both ACP and Data-Plane, the ACP Edge node needs to de-multiplex packets from NMS hosts into ACP VRF and Data-Plane. This is sometimes called "VRF select". If the ACP VRF has no overlapping IPv6 addresses with the Data-Plane (it should have no overlapping addresses), then this function can use the IPv6 Destination address. The problem is Source Address Selection on the NMS Host(s) according to RFC6724.

Consider the simple case: The ACP uses only one ULA prefix, the ACP IPv6 prefix for the Combined ACP and Data-Plane interface is covered by that ULA prefix. The ACP edge node announces both the ACP IPv6 prefix and one (or more) prefixes for the Data-Plane. Without further policy configurations on the NMS Host(s), it may select its ACP address as a source address for Data-Plane ULA destinations because of Rule 8 of RFC6724. The ACP edge node can pass on the packet to the Data-Plane, but the ACP source address should not be used for Data-Plane traffic, and return traffic may fail.
If the ACP carries multiple ULA prefixes or non-ULA ACP connect prefixes, then the correct source address selection becomes even more problematic.

With separate ACP connect and Data-Plane subnets and RFC4191 prefix announcements that are to be routed across the ACP connect interface, RFC6724 source address selection Rule 5 (use address of outgoing interface) will be used, so that above problems do not occur, even in more complex cases of multiple ULA and non-ULA prefixes in the ACP routing table.

To achieve the same behavior with a Combined ACP and Data-Plane interface, the ACP Edge Node needs to behave as two separate routers on the interface: One link-local IPv6 address/router for its ACP reachability, and one link-local IPv6 address/router for its Data-Plane reachability. The Router Advertisements for both are as described above (Section 8.1.3): For the ACP, the ACP prefix is announced together with RFC4191 option for the prefixes routed across the ACP and lifetime=0 to disqualify this next-hop as a default router. For the Data-Plane, the Data-Plane prefix(es) are announced together with whatever default router parameters are used for the Data-Plane.

In result, RFC6724 source address selection Rule 5.5 may result in the same correct source address selection behavior of NMS hosts without further configuration on it as the separate ACP connect and Data-Plane interfaces. As described in the text for Rule 5.5, this is only a MAY, because IPv6 hosts are not required to track next-hop information. If an NMS Host does not do this, then separate ACP connect and Data-Plane interfaces are the preferable method of attachment. Hosts implementing [RFC8028] should (instead of may) implement [RFC6724] Rule 5.5, so it is preferred for hosts to support [RFC8028].

ACP edge nodes MAY support the Combined ACP and Data-Plane interface.

8.1.5.  Use of GRASP

GRASP can and should be possible to use across ACP connect interfaces, especially in the architectural correct solution when it is used as a mechanism to connect Software (e.g., ASA or legacy NMS applications) to the ACP. Given how the ACP is the security and transport substrate for GRASP, the trustworthiness of nodes/software allowed to participate in the ACP GRASP domain is one of the main reasons why the ACP section describes no solution with non-ACP routers participating in the ACP routing table.
ACP connect interfaces can be dealt with in the GRASP ACP domain the same as any other ACP interface assuming that any physical ACP connect interface is physically protected from attacks and that the connected Software or NMS Hosts are equally trusted as that on other ACP nodes. ACP edge nodes SHOULD have options to filter GRASP messages in and out of ACP connect interfaces (permit/deny) and MAY have more fine-grained filtering (e.g., based on IPv6 address of originator or objective).

When using "Combined ACP and Data-Plane Interfaces", care must be taken that only GRASP messages intended for the ACP GRASP domain received from Software or NMS Hosts are forwarded by ACP edge nodes. Currently there is no definition for a GRASP security and transport substrate beside the ACP, so there is no definition how such Software/NMS Host could participate in two separate GRASP Domains across the same subnet (ACP and Data-Plane domains). At current it is assumed that all GRASP packets on a Combined ACP and Data-Plane interface belong to the GRASP ACP Domain. They must all use the ACP IPv6 addresses of the Software/NMS Hosts. The link-local IPv6 addresses of Software/NMS Hosts (used for GRASP M_DISCOVERY and M_FLOOD messages) are also assumed to belong to the ACP address space.

8.2. ACP through Non-ACP L3 Clouds (Remote ACP neighbors)

Not all nodes in a network may support the ACP. If non-ACP Layer-2 devices are between ACP nodes, the ACP will work across it since it is IP based. However, the autonomic discovery of ACP neighbors via DULL GRASP is only intended to work across L2 connections, so it is not sufficient to autonomically create ACP connections across non-ACP Layer-3 devices.

8.2.1. Configured Remote ACP neighbor

On the ACP node, remote ACP neighbors are configured explicitly. The parameters of such a "connection" are described in the following ABNF.

```
connection = [ method , local-addr, remote-addr, ?pmtu ]
method = [ "IKEv2", ?port ]
method //= [ "DTLS", port ]
local-addr = [ address, ?vrf ]
remote-addr = [ address ]
address = { "any" | ipv4-address | ipv6-address }
vrf = tstr ; Name of a VRF on this node with local-address
```

Figure 16: Parameters for remote ACP neighbors
Explicit configuration of a remote-peer according to this ABNF provides all the information to build a secure channel without requiring a tunnel to that peer and running DULL GRASP inside of it.

The configuration includes the parameters otherwise signaled via DULL GRASP: local address, remote (peer) locator and method. The differences over DULL GRASP local neighbor discovery and secure channel creation are as follows:

- The local and remote address can be IPv4 or IPv6 and are typically global scope addresses.

- The VRF across which the connection is built (and in which local-addr exists) can to be specified. If vrf is not specified, it is the default VRF on the node. In DULL GRASP the VRF is implied by the interface across which DULL GRASP operates.

- If local address is "any", the local address used when initiating a secure channel connection is decided by source address selection ([RFC6724] for IPv6). As a responder, the connection listens on all addresses of the node in the selected VRF.

- Configuration of port is only required for methods where no defaults exist (e.g., "DTLS").

- If remote address is "any", the connection is only a responder. It is a "hub" that can be used by multiple remote peers to connect simultaneously - without having to know or configure their addresses. Example: Hub site for remote "spoke" sites reachable over the Internet.

- Pmtu should be configurable to overcome issues/limitations of Path MTU Discovery (PMTUD).

- IKEv2/IPsec to remote peers should support the optional NAT Traversal (NAT-T) procedures.

8.2.2. Tunneled Remote ACP Neighbor

An IPinIP, GRE or other form of pre-existing tunnel is configured between two remote ACP peers and the virtual interfaces representing the tunnel are configured for "ACP enable". This will enable IPv6 link local addresses and DULL on this tunnel. In result, the tunnel is used for normal "L2 adjacent" candidate ACP neighbor discovery with DULL and secure channel setup procedures described in this document.
Tunneled Remote ACP Neighbor requires two encapsulations: the configured tunnel and the secure channel inside of that tunnel. This makes it in general less desirable than Configured Remote ACP Neighbor. Benefits of tunnels are that it may be easier to implement because there is no change to the ACP functionality - just running it over a virtual (tunnel) interface instead of only native interfaces. The tunnel itself may also provide PMTUD while the secure channel method may not. Or the tunnel mechanism is permitted/possible through some firewall while the secure channel method may not.

8.2.3. Summary

Configured/Tunneled Remote ACP neighbors are less "indestructible" than L2 adjacent ACP neighbors based on link local addressing, since they depend on more correct Data-Plane operations, such as routing and global addressing.

Nevertheless, these options may be crucial to incrementally deploy the ACP, especially if it is meant to connect islands across the Internet. Implementations SHOULD support at least Tunneled Remote ACP Neighbors via GRE tunnels - which is likely the most common router-to-router tunneling protocol in use today.

9. Benefits (Informative)

9.1. Self-Healing Properties

The ACP is self-healing:

- New neighbors will automatically join the ACP after successful validation and will become reachable using their unique ULA address across the ACP.

- When any changes happen in the topology, the routing protocol used in the ACP will automatically adapt to the changes and will continue to provide reachability to all nodes.

- The ACP tracks the validity of peer certificates and tears down ACP secure channels when a peer certificate has expired. When short-lived certificates with lifetimes in the order of OCSP/CRL refresh times are used, then this allows for removal of invalid peers (whose certificate was not renewed) at similar speeds as when using OCSP/CRL. The same benefit can be achieved when using CRL/OCSP, periodically refreshing the revocation information and also tearing down ACP secure channels when the peers (long-lived) certificate is revoked. There is no requirement against ACP implementations to require this enhancement though to keep the mandatory implementations simpler.
The ACP can also sustain network partitions and mergers. Practically all ACP operations are link local, where a network partition has no impact. Nodes authenticate each other using the domain certificates to establish the ACP locally. Addressing inside the ACP remains unchanged, and the routing protocol inside both parts of the ACP will lead to two working (although partitioned) ACPs.

There are few central dependencies: A certificate revocation list (CRL) may not be available during a network partition; a suitable policy to not immediately disconnect neighbors when no CRL is available can address this issue. Also, an ACP registrar or Certificate Authority might not be available during a partition. This may delay renewal of certificates that are to expire in the future, and it may prevent the enrollment of new nodes during the partition.

Highly resilient ACP designs can be built by using ACP registrars with embedded sub-CA, as outlined in Section 10.2.4. As long as a partition is left with one or more of such ACP registrars, it can continue to enroll new candidate ACP nodes as long as the ACP registrars sub-CA certificate does not expire. Because the ACP addressing relies on unique Registrar-IDs, a later re-merge of partitions will also not cause problems with ACP addresses assigned during partitioning.

After a network partition, a re-merge will just establish the previous status, certificates can be renewed, the CRL is available, and new nodes can be enrolled everywhere. Since all nodes use the same trust anchor(s), a re-merge will be smooth.

Merging two networks with different trust anchors requires the ACP nodes to trust the union of Trust Anchors. As long as the routing-subdomain hashes are different, the addressing will not overlap, except for the low probability of a 40-bit hash collision in SHA256 (see Section 6.10). Note that the complete mechanisms to merge networks is out of scope of this specification.

It is also highly desirable for implementation of the ACP to be able to run it over interfaces that are administratively down. If this is not feasible, then it might instead be possible to request explicit operator override upon administrative actions that would administratively bring down an interface across which the ACP is running. Especially if bringing down the ACP is known to disconnect the operator from the node. For example any such down administrative action could perform a dependency check to see if the transport connection across which this action is performed is affected by the down action (with default RPL routing used, packet forwarding will be symmetric, so this is actually possible to check).
9.2. Self-Protection Properties

9.2.1. From the outside

As explained in Section 6, the ACP is based on secure channels built between nodes that have mutually authenticated each other with their domain certificates. The channels themselves are protected using standard encryption technologies such as DTLS or IPsec which provide additional authentication during channel establishment, data integrity and data confidentiality protection of data inside the ACP and in addition, provide replay protection.

An attacker will not be able to join the ACP unless having a valid domain certificate, also packet injection and sniffing traffic will not be possible due to the security provided by the encryption protocol.

The ACP also serves as protection (through authentication and encryption) for protocols relevant to OAM that may not have secured protocol stack options or where implementation or deployment of those options fail on some vendor/product/customer limitations. This includes protocols such as SNMP ([RFC3411]), NTP ([RFC5905]), PTP ([IEEE-1588-2008]), DNS ([RFC1886]), DHCPv6 ([RFC3315]), syslog ([RFC3164]), Radius ([RFC2865]), Diameter ([RFC6733]), TACACS ([RFC1492]), IPFIX ([RFC7011]), Netflow ([RFC3954]) - just to name a few. Protection via the ACP secure hop-by-hop channels for these protocols is meant to be only a stopgap though: The ultimate goal is for these and other protocols to use end-to-end encryption utilizing the domain certificate and rely on the ACP secure channels primarily for zero-touch reliable connectivity, but not primarily for security.

The remaining attack vector would be to attack the underlying ACP protocols themselves, either via directed attacks or by denial-of-service attacks. However, as the ACP is built using link-local IPv6 addresses, remote attacks from the data-plane are impossible as long as the data-plane has no facilities to remotely sent IPv6 link-local packets. The only exception are ACP connected interfaces which require higher physical protection. The ULA addresses are only reachable inside the ACP context, therefore, unreachable from the Data-Plane. Also, the ACP protocols should be implemented to be attack resistant and not consume unnecessary resources even while under attack.

9.2.2. From the inside

The security model of the ACP is based on trusting all members of the group of nodes that receive an ACP domain certificate for the same
domain. Attacks from the inside by a compromised group member are therefore the biggest challenge.

Group members must be protected against attackers so that there is no easy way to compromise them, or use them as a proxy for attacking other devices across the ACP. For example, management plane functions (transport ports) should only be reachable from the ACP but not the Data-Plane. Especially for those management plane functions that have no good protection by themselves because they do not have secure end-to-end transport and to whom ACP does not only provides automatic reliable connectivity but also protection against attacks. Protection across all potential attack vectors is typically easier to do in devices whose software is designed from the ground up with security in mind than with legacy software based systems where the ACP is added on as another feature.

As explained above, traffic across the ACP SHOULD still be end-to-end encrypted whenever possible. This includes traffic such as GRASP, EST and BRSKI inside the ACP. This minimizes man in the middle attacks by compromised ACP group members. Such attackers cannot eavesdrop or modify communications, they can just filter them (which is unavoidable by any means).

See Appendix A.10.8 for further considerations how to avoid and deal with compromised nodes.

9.3. The Administrator View

An ACP is self-forming, self-managing and self-protecting, therefore has minimal dependencies on the administrator of the network. Specifically, since it is (intended to be) independent of configuration, there is no scope for configuration errors on the ACP itself. The administrator may have the option to enable or disable the entire approach, but detailed configuration is not possible. This means that the ACP must not be reflected in the running configuration of nodes, except a possible on/off switch (and even that is undesirable).

While configuration is not possible, an administrator must have full visibility of the ACP and all its parameters, to be able to do trouble-shooting. Therefore, an ACP must support all show and debug options, as for any other network function. Specifically, a network management system or controller must be able to discover the ACP, and monitor its health. This visibility of ACP operations must clearly be separated from visibility of Data-Plane so automated systems will never have to deal with ACP aspect unless they explicitly desire to do so.
Since an ACP is self-protecting, a node not supporting the ACP, or without a valid domain certificate cannot connect to it. This means that by default a traditional controller or network management system cannot connect to an ACP. See Section 8.1.1 for more details on how to connect an NMS host into the ACP.

10. ACP Operations (Informative)

The following sections document important operational aspects of the ACP. They are not normative because they do not impact the interoperability between components of the ACP, but they include recommendations/requirements for the internal operational model beneficial or necessary to achieve the desired use-case benefits of the ACP (see Section 3).

- Section 10.1 describes recommended operator diagnostics capabilities of ACP nodes. They have been derived from diagnostic of a commercially available ACP implementation.

- Section 10.2 describes high level how an ACP registrar needs to work, what its configuration parameters are and specific issues impacting the choices of deployment design due to renewal and revocation issues. It describes a model where ACP Registrars have their own sub-CA to provide the most distributed deployment option for ACP Registrars, and it describes considerations for centralized policy control of ACP Registrar operations.

- Section 10.3 describes suggested ACP node behavior and operational interfaces (configuration options) to manage the ACP in so-called greenfield devices (previously unconfigured) and brownfield devices (preconfigured).

The recommendations and suggestions of this chapter were derived from operational experience gained with a commercially available pre-standard ACP implementation.

10.1. ACP (and BRSKI) Diagnostics

Even though ACP and ANI in general are taking out many manual configuration mistakes through their automation, it is important to provide good diagnostics for them.

The basic diagnostics is support of (yang) data models representing the complete (auto-)configuration and operational state of all components: BRSKI, GRASP, ACP and the infrastructure used by them: TLS/DTLS, IPsec, certificates, trust anchors, time, VRF and so on. While necessary, this is not sufficient:
Simply representing the state of components does not allow operators to quickly take action — unless they do understand how to interpret the data, and that can mean a requirement for deep understanding of all components and how they interact in the ACP/ANI.

Diagnostic supports should help to quickly answer the questions operators are expected to ask, such as "is the ACP working correctly?", or "why is there no ACP connection to a known neighboring node?"

In current network management approaches, the logic to answer these questions is most often built as centralized diagnostics software that leverages the above mentioned data models. While this approach is feasible for components utilizing the ANI, it is not sufficient to diagnose the ANI itself:

- Developing the logic to identify common issues requires operational experience with the components of the ANI. Letting each management system define its own analysis is inefficient.
- When the ANI is not operating correctly, it may not be possible to run diagnostics from remote because of missing connectivity. The ANI should therefore have diagnostic capabilities available locally on the nodes themselves.
- Certain operations are difficult or impossible to monitor in real-time, such as initial bootstrap issues in a network location where no capabilities exist to attach local diagnostics. Therefore it is important to also define means of capturing (logging) diagnostics locally for later retrieval. Ideally, these captures are also non-volatile so that they can survive extended power-off conditions — for example when a device that fails to be brought up zero-touch is being sent back for diagnostics at a more appropriate location.

The most simple form of diagnostics answering questions such as the above is to represent the relevant information sequentially in dependency order, so that the first non-expected/non-operational item is the most likely root cause. Or just log/highlight that item. For example:

Q: Is ACP operational to accept neighbor connections:

- Check if any potentially necessary configuration to make ACP/ANI operational are correct (see Section 10.3 for a discussion of such commands).
o Does the system time look reasonable, or could it be the default system time after clock chip battery failure (certificate checks depend on reasonable notion of time).

o Does the node have keying material - domain certificate, trust anchors.

o If no keying material and ANI is supported/enabled, check the state of BR SKI (not detailed in this example).

o Check the validity of the domain certificate:
  * Does the certificate authenticate against the trust anchor?
  * Has it been revoked?
  * Was the last scheduled attempt to retrieve a CRL successful (e.g., do we know that our CRL information is up to date).
  * Is the certificate valid: validity start time in the past, expiration time in the future?
  * Does the certificate have a correctly formatted ACP domain information field?

o Was the ACP VRF successfully created?

o Is ACP enabled on one or more interfaces that are up and running?

If all this looks good, the ACP should be running locally "fine" - but we did not check any ACP neighbor relationships.

Question: why does the node not create a working ACP connection to a neighbor on an interface?

o Is the interface physically up? Does it have an IPv6 link-local address?

o Is it enabled for ACP?

o Do we successfully send DULL GRASP messages to the interface (link layer errors)?

o Do we receive DULL GRASP messages on the interface? If not, some intervening L2 equipment performing bad MLD snooping could have caused problems. Provide e.g., diagnostics of the MLD querier IPv6 and MAC address.
o Do we see the ACP objective in any DULL GRASP message from that interface? Diagnose the supported secure channel methods.

o Do we know the MAC address of the neighbor with the ACP objective? If not, diagnose SLAAC/ND state.

o When did we last attempt to build an ACP secure channel to the neighbor?

o If it failed, why:

* Did the neighbor close the connection on us or did we close the connection on it because the domain certificate membership failed?

* If the neighbor closed the connection on us, provide any error diagnostics from the secure channel protocol.

* If we failed the attempt, display our local reason:

  + There was no common secure channel protocol supported by the two neighbors (this could not happen on nodes supporting this specification because it mandates common support for IPsec).

  + The ACP domain certificate membership check (Section 6.1.2) fails:

    - The neighbors certificate does not have the required trust anchor. Provide diagnostics which trust anchor it has (can identify whom the device belongs to).

    - The neighbors certificate does not have the same domain (or no domain at all). Diagnose domain-name and potentially other cert info.

    - The neighbors certificate has been revoked or could not be authenticated by OCSP.

    - The neighbors certificate has expired - or is not yet valid.

* Any other connection issues in e.g., IKEv2 / IPsec, DTLS?.

Question: Is the ACP operating correctly across its secure channels?

o Are there one or more active ACP neighbors with secure channels?
Is the RPL routing protocol for the ACP running?

Is there a default route to the root in the ACP routing table?

Is there for each direct ACP neighbor not reachable over the ACP virtual interface to the root a route in the ACP routing table?

Is ACP GRASP running?

Is at least one SRV.est objective cached (to support certificate renewal)?

Is there at least one BRSKI registrar objective cached (in case BRSKI is supported)

Is BRSKI proxy operating normally on all interfaces where ACP is operating?

... 

These lists are not necessarily complete, but illustrate the principle and show that there are variety of issues ranging from normal operational causes (a neighbor in another ACP domain) over problems in the credentials management (certificate lifetimes), explicit security actions (revocation) or unexpected connectivity issues (intervening L2 equipment).

The items so far are illustrating how the ANI operations can be diagnosed with passive observation of the operational state of its components including historic/cached/counted events. This is not necessary sufficient to provide good enough diagnostics overall:

The components of ACP and BRSKI are designed with security in mind but they do not attempt to provide diagnostics for building the network itself. Consider two examples:

1. BRSKI does not allow for a neighboring device to identify the pledges certificate (IDevID). Only the selected BRSKI registrar can do this, but it may be difficult to disseminate information about undesired pledges from those BRSKI registrars to locations/nodes where information about those pledges is desired.

2. The Link Layer Discovery Protocol (LLDP, [LLDP]) disseminates information about nodes to their immediate neighbors, such as node model/type/software and interface name/number of the connection. This information is often helpful or even necessary in network diagnostics. It can equally considered to be too
insecure to make this information available unprotected to all
possible neighbors.

An "interested adjacent party" can always determine the IDevID of a
BRSKI pledge by behaving like a BRSKI proxy/registrar. Therefore the
IDevID of a BRSKI pledge is not meant to be protected - it just has
to be queried and is not signaled unsolicited (as it would be in
LLDP) so that other observers on the same subnet can determine who is
an "interested adjacent party".

10.2. ACP Registrars

As described in Section 6.10.7, the ACP addressing mechanism is
designed to enable lightweight, distributed and uncoordinated ACP
registrars that are providing ACP address prefixes to candidate ACP
nodes by enrolling them with an ACP domain certificate into an ACP
domain via any appropriate mechanism/protocol, automated or not.

This section discusses informatively more details and options for ACP
registrars.

10.2.1. Registrar interactions

This section summarizes and discusses the interactions with other
entities required by an ACP registrar.

In a simple instance of an ACP network, no central NOC component
beside a trust anchor (root CA) is required. One or more
uncoordinated acting ACP registrar can be set up, performing the
following interactions:

To orchestrate enrolling a candidate ACP node autonomically, the ACP
registrar can rely on the ACP and use Proxies to reach the candidate
ACP node, therefore allowing minimum pre-existing (auto-)configured
network services on the candidate ACP node. BRSKI defines the BRSKI
proxy, a design that can be adopted for various protocols that
Pledges/candidate ACP nodes could want to use, for example BRSKI over
CoAP (Constrained Application Protocol), or proxying of Netconf.

To reach a trust anchor unaware of the ACP, the ACP registrar would
use the Data-Plane. ACP and Data-Plane in an ACP registrar could
(and by default should be) completely isolated from each other at the
network level. Only applications such as the ACP registrar would
need the ability for their transport stacks to access both.

In non-autonomic enrollment options, the Data-Plane between a ACP
registrar and the candidate ACP node needs to be configured first.
This includes the ACP registrar and the candidate ACP node. Then any
appropriate set of protocols can be used between ACP registrar and candidate ACP node to discover the other side, and then connect and enroll (configure) the candidate ACP node with an ACP domain certificate. Netconf ZeroTouch ([I-D.ietf-netconf-zerotouch]) is an example protocol that could be used for this. BRSKI using optional discovery mechanisms is equally a possibility for candidate ACP nodes attempting to be enrolled across non-ACP networks, such as the Internet.

When candidate ACP nodes have secure bootstrap, such as BRSKI Pledges, they will not trust to be configured/enrolled across the network, unless being presented with a voucher (see [RFC8366]) authorizing the network to take possession of the node. An ACP registrar will then need a method to retrieve such a voucher, either offline, or online from a MASA (Manufacturer Authorized Signing Authority). BRSKI and Netconf ZeroTouch are two protocols that include capabilities to present the voucher to the candidate ACP node.

An ACP registrar could operate EST for ACP certificate renewal and/or act as a CRL Distribution point. A node performing these services does not need to support performing (initial) enrollment, but it does require the same above described connectivity as an ACP registrar: via the ACP to ACP nodes and via the Data-Plane to the trust anchor and other sources of CRL information.

10.2.2. Registrar Parameter

The interactions of an ACP registrar outlined Section 6.10.7 and Section 10.2.1 above depend on the following parameters:

A URL to the trust anchor (root CA) and credentials so that the ACP registrar can let the trust anchor sign candidate ACP member certificates.

The ACP domain-name.

The Registrar-ID to use. This could default to a MAC address of the ACP registrar.

For recovery, the next-useable Node-IDs for zone (Zone-ID=0) sub-addressing scheme, for Vlong /112 and for Vlong /1120 sub-addressing scheme. These IDs would only need to be provisioned after recovering from a crash. Some other mechanism would be required to remember these IDs in a backup location or to recover them from the set of currently known ACP nodes.
Policies if candidate ACP nodes should receive a domain certificate or not, for example based on the devices LDevID as in BRSKI. The ACP registrar may have a whitelist or blacklist of devices serialNumbers from their LDevID.

Policies what type of address prefix to assign to a candidate ACP devices, based on likely the same information.

For BRSKI or other mechanisms using vouchers: Parameters to determine how to retrieve vouchers for specific type of secure bootstrap candidate ACP nodes (such as MASA URLs), unless this information is automatically learned such as from the IDevID of candidate ACP nodes (as defined in BRSKI).

10.2.3. Certificate renewal and limitations

When an ACP node renews/rekeys its certificate, it may end up doing so via a different registrar (e.g., EST server) than the one it originally received its ACP domain certificate from, for example because that original ACP registrar is gone. The ACP registrar through which the renewal/rekeying is performed would by default trust the ACP domain information from the ACP nodes current ACP domain certificate and maintain this information so that the ACP node maintains its ACP address prefix. In EST renewal/rekeying, the ACP nodes current ACP domain certificate is signaled during the TLS handshake.

This simple scenario has two limitations:

1. The ACP registrars cannot directly assign certificates to nodes and therefore needs an "online" connection to the trust anchor (root CA).

2. Recovery from a compromised ACP registrar is difficult. When an ACP registrar is compromised, it can insert for example conflicting ACP domain information and create thereby an attack against other ACP nodes through the ACP routing protocol.

Even when such a malicious ACP registrar is detected, resolving the problem may be difficult because it would require identifying all the wrong ACP domain certificates assigned via the ACP registrar after it was compromised. And without additional centralized tracking of assigned certificates there is no way to do this.
10.2.4. ACP Registrars with sub-CA

In situations, where either of the above two limitations are an issue, ACP registrars could also be sub-CAs. This removes the need for connectivity to a root-CA whenever an ACP node is enrolled, and reduces the need for connectivity of such an ACP registrar to a root-CA to only those times when it needs to renew its own certificate. The ACP registrar would also now use its own (sub-CA) certificate to enroll and sign the ACP nodes certificates, and therefore it is only necessary to revoke a compromised ACP registrars sub-CA certificate. Alternatively one can let it expire and not renew it, when the certificate of the sub-CA is appropriately short-lived.

As the ACP domain membership check verifies a peer ACP node’s ACP domain certificate trust chain, it will also verify the signing certificate which is the compromised/revoked sub-CA certificate. Therefore ACP domain membership for an ACP node enrolled from a compromised and discovered ACP registrar will fail.

ACP nodes enrolled by a compromised ACP registrar would automatically fail to establish ACP channels and ACP domain certificate renewal via EST and therefore revert to their role as a candidate ACP members and attempt to get a new ACP domain certificate from an ACP registrar - for example, via BRSKI. In result, ACP registrars that have an associated sub-CA makes isolating and resolving issues with compromised registrars easier.

Note that ACP registrars with sub-CA functionality also can control the lifetime of ACP domain certificates easier and therefore also be used as a tool to introduce short lived certificates and not rely on CRL, whereas the certificates for the sub-CAs themselves could be longer lived and subject to CRL.

10.2.5. Centralized Policy Control

When using multiple, uncoordinated ACP registrars, several advanced operations are potentially more complex than with a single, resilient policy control backend, for example including but not limited to:

- Which candidate ACP node is permitted or not permitted into an ACP domain. This may not be a decision to be taken upfront, so that a per-serialNumber policy can be loaded into every ACP registrar. Instead, it may better be decided in real-time including potentially a human decision in a NOC.

- Tracking of all enrolled ACP nodes and their certificate information. For example in support of revoking individual ACP nodes certificates.
More flexible policies what type of address prefix or even what specific address prefix to assign to a candidate ACP node.

These and other operations could be introduced more easily by introducing a centralized Policy Management System (PMS) and modifying ACP registrar behavior so that it queries the PMS for any policy decision occurring during the candidate ACP node enrollment process and/or the ACP node certificate renewal process. For example, which ACP address prefix to assign. Likewise the ACP registrar would report any relevant state change information to the PMS as well, for example when a certificate was successfully enrolled onto a candidate ACP node.

10.3. Enabling and disabling ACP/ANI

Both ACP and BRSKI require interfaces to be operational enough to support sending/receiving their packets. In node types where interfaces are by default (e.g., without operator configuration) enabled, such as most L2 switches, this would be less of a change in behavior than in most L3 devices (e.g.: routers), where interfaces are by default disabled. In almost all network devices it is common though for configuration to change interfaces to a physically disabled state and that would break the ACP.

In this section, we discuss a suggested operational model to enable/disable interfaces and nodes for ACP/ANI in a way that minimizes the risk of operator action to break the ACP in this way, and that also minimizes operator surprise when ACP/ANI becomes supported in node software.

10.3.1. Filtering for non-ACP/ANI packets

Whenever this document refers to enabling an interface for ACP (or BRSKI), it only requires to permit the interface to send/receive packets necessary to operate ACP (or BRSKI) - but not any other Data-Plane packets. Unless the Data-Plane is explicitly configured/enabled, all packets not required for ACP/BRSKI should be filtered on input and output:

Both BRSKI and ACP require link-local only IPv6 operations on interfaces and DULL GRASP. IPv6 link-local operations means the minimum signaling to auto-assign an IPv6 link-local address and talk to neighbors via their link-local address: SLAAC (Stateless Address Auto-Configuration - [RFC4862]) and ND (Neighbor Discovery - [RFC4861]). When the device is a BRSKI pledge, it may also require TCP/TLS connections to BRSKI proxies on the interface. When the device has keying material, and the ACP is running, it requires DULL GRASP packets and packets necessary for the secure-channel mechanism.
it supports, e.g., IKEv2 and IPsec ESP packets or DTLS packets to the IPv6 link-local address of an ACP neighbor on the interface. It also requires TCP/TLS packets for its BRSKI proxy functionality, if it does support BRSKI.

10.3.2.  Admin Down State

Interfaces on most network equipment have at least two states: "up" and "down". These may have product specific names. "down" for example could be called "shutdown" and "up" could be called "no shutdown". The "down" state disables all interface operations down to the physical level. The "up" state enables the interface enough for all possible L2/L3 services to operate on top of it and it may also auto-enable some subset of them. More commonly, the operations of various L2/L3 services is controlled via additional node-wide or interface level options, but they all become only active when the interface is not "down". Therefore an easy way to ensure that all L2/L3 operations on an interface are inactive is to put the interface into "down" state. The fact that this also physically shuts down the interface is in many cases just a side effect, but it may be important in other cases (see below, Section 10.3.2.2).

To provide ACP/ANI resilience against operators configuring interfaces to "down" state, this document recommends to separate the "down" state of interfaces into an "admin down" state where the physical layer is kept running and ACP/ANI can use the interface and a "physical down" state. Any existing "down" configurations would map to "admin down". In "admin down", any existing L2/L3 services of the Data-Plane should see no difference to "physical down" state. To ensure that no Data-Plane packets could be sent/received, packet filtering could be established automatically as described above in Section 10.3.1.

As necessary (see discussion below) new configuration options could be introduced to issue "physical down". The options should be provided with additional checks to minimize the risk of issuing them in a way that breaks the ACP without automatic restoration. For example they could be denied to be issued from a control connection (netconf/ssh) that goes across the interface itself ("do not disconnect yourself"). Or they could be performed only temporary and only be made permanent with additional later reconfirmation.

In the following sub-sections important aspects to the introduction of "admin down" state are discussed.
10.3.2.1. Security

Interfaces are physically brought down (or left in default down state) as a form of security. "Admin down" state as described above provides also a high level of security because it only permits ACP/ANI operations which are both well secured. Ultimately, it is subject to security review for the deployment whether "admin down" is a feasible replacement for "physical down".

The need to trust the security of ACP/ANI operations needs to be weighed against the operational benefits of permitting this: Consider the typical example of a CPE (customer premises equipment) with no on-site network expert. User ports are in physical down state unless explicitly configured not to be. In a misconfiguration situation, the uplink connection is incorrectly plugged into such as user port. The device is disconnected from the network and therefore no diagnostics from the network side is possible anymore. Alternatively, all ports default to "admin down". The ACP (but not the Data-Plane) would still automatically form. Diagnostics from the network side is possible and operator reaction could include to either make this port the operational uplink port or to instruct re-cabling. Security wise, only ACP/ANI could be attacked, all other functions are filtered on interfaces in "admin down" state.

10.3.2.2. Fast state propagation and Diagnostics

"Physical down" state propagates on many interface types (e.g., Ethernet) to the other side. This can trigger fast L2/L3 protocol reaction on the other side and "admin down" would not have the same (fast) result.

Bringing interfaces to "physical down" state is to the best of our knowledge always a result of operator action, but today, never the result of (autonomic) L2/L3 services running on the nodes. Therefore one option is to change the operator action to not rely on link-state propagation anymore. This may not be possible when both sides are under different operator control, but in that case it is unlikely that the ACP is running across the link and actually putting the interface into "physical down" state may still be a good option.

Ideally, fast physical state propagation is replaced by fast software driven state propagation. For example a DULL GRASP "admin-state" objective could be used to auto configure a Bidirectional Forwarding Protocol (BFD, [RFC5880]) session between the two sides of the link that would be used to propagate the "up" vs. admin down state.

Triggering physical down state may also be used as a mean of diagnosing cabling in the absence of easier methods. It is more
complex than automated neighbor diagnostics because it requires
coordinated remote access to both (likely) sides of a link to
determine whether up/down toggling will cause the same reaction on
the remote side.

See Section 10.1 for a discussion about how LLDP and/or diagnostics
via GRASP could be used to provide neighbor diagnostics, and
therefore hopefully eliminating the need for "physical down" for
neighbor diagnostics - as long as both neighbors support ACP/ANI.

10.3.2.3. Low Level Link Diagnostics

"Physical down" is performed to diagnose low-level interface behavior
when higher layer services (e.g., IPv6) are not working. Especially
Ethernet links are subject to a wide variety of possible wrong
configuration/cablings if they do not support automatic selection of
variable parameters such as speed (10/100/1000 Mbps), crossover
(Auto-MDIX) and connector (fiber, copper - when interfaces have
multiple but can only enable one at a time). The need for low level
link diagnostic can therefore be minimized by using fully auto
configuring links.

In addition to "Physical down", low level diagnostics of Ethernet or
other interfaces also involve the creation of other states on
interfaces, such as physical Loopback (internal and/or external) or
bringing down all packet transmissions for reflection/cable-length
measurements. Any of these options would disrupt ACP as well.

In cases where such low-level diagnostics of an operational link is
desired but where the link could be a single point of failure for the
ACP, ASA on both nodes of the link could perform a negotiated
diagnostics that automatically terminates in a predetermined manner
without dependence on external input ensuring the link will become
operational again.

10.3.2.4. Power Consumption Issues

Power consumption of "physical down" interfaces, may be significantly
lower than those in "admin down" state, for example on long-range
fiber interfaces. Bringing up interfaces, for example to probe
reachability, may also consume additional power. This can make these
type of interfaces inappropriate to operate purely for the ACP when
they are not currently needed for the Data-Plane.
10.3.3. Interface level ACP/ANI enable

The interface level configuration option "ACP enable" enables ACP operations on an interface, starting with ACP neighbor discovery via DULL GRAP. The interface level configuration option "ANI enable" on nodes supporting BRSKI and ACP starts with BRSKI pledge operations when there is no domain certificate on the node. On ACP/BRSKI nodes, "ACP enable" may not need to be supported, but only "ANI enable". Unless overridden by global configuration options (see later), "ACP/ANI enable" will result in "down" state on an interface to behave as "admin down".

10.3.4. Which interfaces to auto-enable?

(Section 6.3) requires that "ACP enable" is automatically set on native interfaces, but not on non-native interfaces (reminder: a native interface is one that exists without operator configuration action such as physical interfaces in physical devices).

Ideally, ACP enable is set automatically on all interfaces that provide access to additional connectivity that allows to reach more nodes of the ACP domain. The best set of interfaces necessary to achieve this is not possible to determine automatically. Native interfaces are the best automatic approximation.

Consider an ACP domain of ACP nodes transitively connected via native interfaces. A Data-Plane tunnel between two of these nodes that are non-adjacent is created and "ACP enable" is set for that tunnel. ACP RPL sees this tunnel as just as a single hop. Routes in the ACP would use this hop as an attractive path element to connect regions adjacent to the tunnel nodes. In result, the actual hop-by-hop paths used by traffic in the ACP can become worse. In addition, correct forwarding in the ACP now depends on correct Data-Plane forwarding config including QoS, filtering and other security on the Data-Plane path across which this tunnel runs. This is the main issue why "ACP/ANI enable" should not be set automatically on non-native interfaces.

If the tunnel would connect two previously disjoint ACP regions, then it likely would be useful for the ACP. A Data-Plane tunnel could also run across nodes without ACP and provide additional connectivity for an already connected ACP network. The benefit of this additional ACP redundancy has to be weighed against the problems of relying on the Data-Plane. If a tunnel connects two separate ACP regions: how many tunnels should be created to connect these ACP regions reliably enough? Between which nodes? These are all standard tunneled network design questions not specific to the ACP, and there are no generic fully automated answers.
Instead of automatically setting "ACP enable" on these type of interfaces, the decision needs to be based on the use purpose of the non-native interface and "ACP enable" needs to be set in conjunction with the mechanism through which the non-native interface is created/configured.

In addition to explicit setting of "ACP/ANI enable", non-native interfaces also need to support configuration of the ACP RPL cost of the link - to avoid the problems of attracting too much traffic to the link as described above.

Even native interfaces may not be able to automatically perform BRSKI or ACP because they may require additional operator input to become operational. Example include DSL interfaces requiring PPPoE credentials or mobile interfaces requiring credentials from a SIM card. Whatever mechanism is used to provide the necessary config to the device to enable the interface can also be expanded to decide on whether or not to set "ACP/ANI enable".

The goal of automatically setting "ACP/ANI enable" on interfaces (native or not) is to eliminate unnecessary "touches" to the node to make its operation as much as possible "zero-touch" with respect to ACP/ANI. If there are "unavoidable touches" such a creating/configuring a non-native interface or provisioning credentials for a native interface, then "ACP/ANI enable" should be added as an option to that "touch". If a wrong "touch" is easily fixed (not creating another high-cost touch), then the default should be not to enable ANI/ACP, and if it is potentially expensive or slow to fix (e.g., parameters on SIM card shipped to remote location), then the default should be to enable ACP/ANI.

10.3.5. Node Level ACP/ANI enable

A node level command "ACP/ANI enable [up-if-only]" enables ACP or ANI on the node (ANI = ACP + BRSKI). Without this command set, any interface level "ACP/ANI enable" is ignored. Once set, ACP/ANI will operate interface where "ACP/ANI enable" is set. Setting of interface level "ACP/ANI enable" is either automatic (default) or explicit through operator action as described in the previous section.

If the option "up-if-only" is selected, the behavior of "down" interfaces is unchanged, and ACP/ANI will only operate on interfaces where "ACP/ANI enable" is set and that are "up". When it is not set, then "down" state of interfaces with "ACP/ANI enable" is modified to behave as "admin down".
10.3.5.1. Brownfield nodes

A "brownfield" node is one that already has a configured Data-Plane.

Executing global "ACP/ANI enable [up-if-only]" on each node is the only command necessary to create an ACP across a network of brownfield nodes once all the nodes have a domain certificate. When BRSKI is used ("ANI enable"), provisioning of the certificates only requires set-up of a single BRSKI registrar node which could also implement a CA for the network. This is the most simple way to introduce ACP/ANI into existing (== brownfield) networks.

The need to explicitly enable ACP/ANI is especially important in brownfield nodes because otherwise software updates may introduce support for ACP/ANI: Automatic enablement of ACP/ANI in networks where the operator does not only not want ACP/ANI but where the operator likely never even heard of it could be quite irritating to the operator. Especially when "down" behavior is changed to "admin down".

Automatically setting "ANI enable" on brownfield nodes where the operator is unaware of it could also be a critical security issue depending on the vouchers used by BRSKI on these nodes. An attacker could claim to be the owner of these devices and create an ACP that the attacker has access/control over. In networks where the operator explicitly wants to enable the ANI this could not happen, because he would create a BRSKI registrar that would discover attack attempts. Nodes requiring "ownership vouchers" would not be subject to that attack. See [I-D.ietf-anima-bootstrapping-keyinfra] for more details. Note that a global "ACP enable" alone is not subject to these type of attacks, because it always depends on some other mechanism first to provision domain certificates into the device.

10.3.5.2. Greenfield nodes

A "greenfield" node is one that did not have any prior configuration.

For greenfield nodes, only "ANI enable" is relevant. If another mechanism than BRSKI is used to (zero-touch) bootstrap a node, then it is up to that mechanism to provision domain certificates and to set global "ACP enable" as desired.

Nodes supporting full ANI functionality set "ANI enable" automatically when they decide that they are greenfield, e.g., that they are powering on from factory condition. They will then put all native interfaces into "admin down" state and start to perform BRSKI pledge functionality - and once a domain certificate is enrolled they automatically enable ACP.
Attempts for BRSKI pledge operations in greenfield state should terminate automatically when another method of configuring the node is used. Methods that indicate some form of physical possession of the device such as configuration via the serial console port could lead to immediate termination of BRSKI, while other parallel auto configuration methods subject to remote attacks might lead to BRSKI termination only after they were successful. Details of this may vary widely over different type of nodes. When BRSKI pledge operation terminates, this will automatically unset "ANI enable" and should terminate any temporarily needed state on the device to perform BRSKI - DULL GRASP, BRSKI pledge and any IPv6 configuration on interfaces.

10.3.6. Undoing ANI/ACP enable

Disabling ANI/ACP by undoing "ACP/ANI enable" is a risk for the reliable operations of the ACP if it can be executed by mistake or unauthorized. This behavior could be influenced through some additional property in the certificate (e.g., in the domain information extension field) subject to future work: In an ANI deployment intended for convenience, disabling it could be allowed without further constraints. In an ANI deployment considered to be critical more checks would be required. One very controlled option would be to not permit these commands unless the domain certificate has been revoked or is denied renewal. Configuring this option would be a parameter on the BRSKI registrar(s). As long as the node did not receive a domain certificate, undoing "ANI/ACP enable" should not have any additional constraints.

10.3.7. Summary

Node-wide "ACP/ANI enable [up-if-only]" commands enable the operation of ACP/ANI. This is only auto-enabled on ANI greenfield devices, otherwise it must be configured explicitly.

If the option "up-if-only" is not selected, interfaces enabled for ACP/ANI interpret "down" state as "admin down" and not "physical down". In "admin-down" all non-ACP/ANI packets are filtered, but the physical layer is kept running to permit ACP/ANI to operate.

(New) commands that result in physical interruption ("physical down", "loopback") of ACP/ANI enabled interfaces should be built to protect continuance or reestablishment of ACP as much as possible.

Interface level "ACP/ANI enable" control per-interface operations. It is enabled by default on native interfaces and has to be configured explicitly on other interfaces.
Disabling "ACP/ANI enable" global and per-interface should have additional checks to minimize undesired breakage of ACP. The degree of control could be a domain wide parameter in the domain certificates.

10.4. Configuration and the ACP (summary)

There is no desirable configuration for the ACP. Instead, all parameters that need to be configured in support of the ACP are limitations of the solution, but they are only needed in cases where not all components are made autonomic. Wherever this is necessary, it will rely on pre-existing mechanisms for configuration such as CLI or YANG ([RFC7950]) data models.

The most important examples of such configuration include:

- When ACP nodes do not support an autonomic way to receive an ACP domain certificate, for example BRSKI, then such certificate needs to be configured via some pre-existing mechanisms outside the scope of this specification. Today, router have typically a variety of mechanisms to do this.

- Certificate maintenance requires PKI functions. Discovery of these functions across the ACP is automated (see Section 6.1.4), but their configuration is is not.

- When non-ACP capable nodes need to be connected to the ACP, the connecting ACP node needs to be configuration to support this according to Section 8.1.

- When devices are not autonomically bootstrapped, explicit configuration to enable the ACP needs to be applied. See Section 10.3.

- When the ACP needs to be extended across interfaces other than L2, the ACP as defined in this document can not autodiscover candidate neighbors automatically. Remove neighbors need to be configured, see Section 8.2.

Once the ACP is operating, any further configuration for the data-plane can be configured more reliably across the ACP itself because the ACP provides addressing and connectivity (routing) independent of the data-plane itself. For this, the configuration methods simply need to also allow to operate across the ACP VRF – netconf, ssh or any other method.

The ACP also provides additional security through its hop-by-hop encryption for any such configuration operations: Some legacy
configuration methods (SNMP, TFTP, HTTP) may not use end-to-end encryption, and most of the end-to-end secured configuration methods still allow for easy passive observation along the path about configuration taking place (transport flows, port numbers, IP addresses).

The ACP can and should equally be used as the transport to configure any of the aforementioned non-autonomic components of the ACP, but in that case, the same caution needs to be exercised as with data-plane configuration without ACP: Misconfiguration may cause the configuring entity to be disconnected from the node it configures - for example when incorrectly unconfiguring a remote ACP neighbor through which the configured ACP node is reached.

11. Security Considerations

After seeding an ACP by configuring at least one ACP registrar with routing-subdomain and a CA, an ACP is self-protecting and there is no need to apply configuration to make it secure (typically the ACP Registrar doubles as EST server for certificate renewal). Its security therefore does not depend on configuration. This does not include workarounds for non-autonomic components as explained in Section 8. See Section 9.2 for details of how the ACP protects itself against attacks from the outside and to a more limited degree from the inside as well.

However, the security of the ACP depends on a number of other factors:

- The usage of domain certificates depends on a valid supporting PKI infrastructure. If the chain of trust of this PKI infrastructure is compromised, the security of the ACP is also compromised. This is typically under the control of the network administrator.

- Every ACP registrar is critical infrastructure that needs to be hardened against attacks similar to a CA. A malicious registrar can enroll enemy pledgees to an ACP network or break ACP routing by duplicate ACP address assignment to pledgees via their ACP domain certificates.

- Security can be compromised by implementation errors (bugs), as in all products.

There is no prevention of source-address spoofing inside the ACP. This implies that if an attacker gains access to the ACP, it can spoof all addresses inside the ACP and fake messages from any other node.
The ACP is designed to enable automation of current network management and future autonomic peer-to-peer/distributed network automation. Any ACP member can send ACP IPv6 packet to other ACP members and announce via ACP GRASP services to all ACP members without dependency against centralized components.

The ACP relies on peer-to-peer authentication and authorization using ACP certificates. This security model is necessary to enable the autonomic ad-hoc any-to-any connectivity between ACP nodes. It provides infrastructure protection through hop by hop authentication and encryption - without relying on third parties. For any services where this complete autonomic peer-to-peer group security model is appropriate, the ACP domain certificate can also be used unchanged. For example for any type of data-plane routing protocol security.

This ACP security model is designed primarily to protect against attack from the outside, but not against attacks from the inside. To protect against spoofing attacks from compromised on-path ACP nodes, end-to-end encryption inside the ACP is used by new ACP signaling: GRASP across the ACP using TLS. The same is expected from any non-legacy services/protocols using the ACP. Because no group-keys are used, there is no risk for impacted nodes to access end-to-end encrypted traffic from other ACP nodes.

Attacks from impacted ACP nodes against the ACP are more difficult than against the data-plane because of the autoconfiguration of the ACP and the absence of configuration options that could be abused that allow to change/break ACP behavior. This is excluding configuration for workaround in support of non-autonomic components.

Mitigation against compromised ACP members is possible through standard automated certificate management mechanisms including revocation and non-renewal of short-lived certificates. In this version of the specification, there are no further optimization of these mechanisms defined for the ACP (but see Appendix A.10.8).

Higher layer service built using ACP domain certificates should not solely rely on undifferentiated group security when another model is more appropriate/more secure. For example central network configuration relies on a security model where only few especially trusted nodes are allowed to configure the data-plane of network nodes (CLIL, Netconf). This can be done through ACP domain certificates by differentiating them and introduce roles. See Appendix A.10.5.

Fundamentally, security depends on avoiding operator and network operations automation mistakes, implementation and architecture. Autonomic approaches such as the ACP largely eliminate operator
mistakes and make it easier to recover from network operations mistakes. Implementation and architectural mistakes are still possible, as in all networking technologies.

Many details of ACP are designed with security in mind and discussed elsewhere in the document:

IPv6 addresses used by nodes in the ACP are covered as part of the node's domain certificate as described in Section 6.1.1. This allows even verification of ownership of a peers IPv6 address when using a connection authenticated with the domain certificate.

The ACP acts as a security (and transport) substrate for GRASP inside the ACP such that GRASP is not only protected by attacks from the outside, but also by attacks from compromised inside attackers - by relying not only on hop-by-hop security of ACP secure channels, but adding end-to-end security for those GRASP messages. See Section 6.8.2.

ACP provides for secure, resilient zero-touch discovery of EST servers for certificate renewal. See Section 6.1.4.

ACP provides extensible, auto-configuring hop-by-hop protection of the ACP infrastructure via the negotiation of hop-by-hop secure channel protocols. See Section 6.5 and Appendix A.6.

The ACP is designed to minimize attacks from the outside by minimizing its dependency against any non-ACP (Data-Plane) operations/configuration on a node. See also Section 6.12.2.

In combination with BRSKI, ACP enables a resilient, fully zero-touch network solution for short-lived certificates that can be renewed or re-enrolled even after unintentional expiry (e.g., because of interrupted connectivity). See Appendix A.2.

Because ACP secure channels can be long lived, but certificates used may be short lived, secure channels, for example built via IPsec need to be terminated when peer certificates expire. See Section 6.7.3.

The ACP is designed to minimize attacks from the outside by minimizing its dependency against any non-ACP (Data-Plane) operations/configuration on a node. See also Section 6.12.2.

12. IANA Considerations

This document defines the "Autonomic Control Plane".
The IANA is requested to register the value "AN_ACP" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.3.

The IANA is requested to register the value "SRV.est" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.1.4.

Explanation: This document chooses the initially strange looking format "SRV.<service-name>" because these objective names would be in line with potential future simplification of the GRASP objective registry. Today, every name in the GRASP objective registry needs to be explicitly allocated with IANA. In the future, this type of objective names could considered to be automatically registered in that registry for the same service for which <service-name> is registered according to [RFC6335]. This explanation is solely informational and has no impact on the requested registration.

The IANA is requested to create an ACP Parameter Registry with currently one registry table - the "ACP Address Type" table.

"ACP Address Type" Table. The value in this table are numeric values 0...3 paired with a name (string). Future values MUST be assigned using the Standards Action policy defined by [RFC8126]. The following initial values are assigned by this document:

0: ACP Zone Addressing Sub-Scheme (ACP RFC Figure 10) / ACP Manual Addressing Sub-Scheme (ACP RFC Section 6.10.4)
1: ACP Vlong Addressing Sub-Scheme (ACP RFC Section 6.10.5)

13. Acknowledgements

This work originated from an Autonomic Networking project at Cisco Systems, which started in early 2010. Many people contributed to this project and the idea of the Autonomic Control Plane, amongst which (in alphabetical order): Ignas Bagdonas, Parag Bhide, Balaji BL, Alex Clemm, Yves Hertoghs, Bruno Klauser, Max Pritikin, Michael Richardson, Ravi Kumar Vadapalli.

Special thanks to Brian Carpenter, Elwyn Davies, Joel Halpern and Sheng Jiang for their thorough reviews and to Pascal Thubert and Michael Richardson to provide the details for the recommendations of the use of RPL in the ACP.

Further input, review or suggestions were received from: Rene Struik, Brian Carpenter, Benoit Claise, William Atwood and Yongkang Zhang.
14. Change log [RFC Editor: Please remove]

14.1. Initial version

First version of this document: draft-behringer-autonomic-control-plane

14.2. draft-behringer-anima-autonomic-control-plane-00

Initial version of the anima document; only minor edits.

14.3. draft-behringer-anima-autonomic-control-plane-01

- Clarified that the ACP should be based on, and support only IPv6.
- Clarified in intro that ACP is for both, between devices, as well as for access from a central entity, such as an NMS.
- Added a section on how to connect an NMS system.
- Clarified the hop-by-hop crypto nature of the ACP.
- Added several references to GDNP as a candidate protocol.
- Added a discussion on network split and merge. Although, this should probably go into the certificate management story longer term.

14.4. draft-behringer-anima-autonomic-control-plane-02

Addresses (numerous) comments from Brian Carpenter. See mailing list for details. The most important changes are:

- Introduced a new section "overview", to ease the understanding of the approach.
- Merged the previous "problem statement" and "use case" sections into a mostly re-written "use cases" section, since they were overlapping.
- Clarified the relationship with draft-ietf-anima-stable-connectivity

14.5. draft-behringer-anima-autonomic-control-plane-03

- Took out requirement for IPv6 --> that’s in the reference doc.
- Added requirement section.
Changed focus: more focus on autonomic functions, not only virtual out-of-band. This goes a bit throughout the document, starting with a changed abstract and intro.

14.6. draft-ietf-anima-autonomic-control-plane-00

No changes; re-submitted as WG document.

14.7. draft-ietf-anima-autonomic-control-plane-01

- Added some paragraphs in addressing section on "why IPv6 only", to reflect the discussion on the list.
- Moved the Data-Plane ACP out of the main document, into an appendix. The focus is now the virtually separated ACP, since it has significant advantages, and isn’t much harder to do.
- Changed the self-creation algorithm: Part of the initial steps go into the reference document. This document now assumes an adjacency table, and domain certificate. How those get onto the device is outside scope for this document.
- Created a new section 6 "workarounds for non-autonomic nodes", and put the previous controller section (5.9) into this new section. Now, section 5 is "autonomic only", and section 6 explains what to do with non-autonomic stuff. Much cleaner now.
- Added an appendix explaining the choice of RPL as a routing protocol.
- Formalized the creation process a bit more. Now, we create a "candidate peer list" from the adjacency table, and form the ACP with those candidates. Also it explains now better that policy (Intent) can influence the peer selection. (section 4 and 5)
- Introduce a section for the capability negotiation protocol (section 7). This needs to be worked out in more detail. This will likely be based on GRASP.
- Introduce a new parameter: ACP tunnel type. And defines it in the IANA considerations section. Suggest GRE protected with IPSec transport mode as the default tunnel type.
- Updated links, lots of small edits.
14.8. draft-ietf-anima-autonomic-control-plane-02

- Added explicitly text for the ACP channel negotiation.
- Merged draft-behringer-anima-autonomic-addressing-02 into this document, as suggested by WG chairs.

14.9. draft-ietf-anima-autonomic-control-plane-03

- Changed Neighbor discovery protocol from GRASP to mDNS. Bootstrap protocol team decided to go with mDNS to discover bootstrap proxy, and ACP should be consistent with this. Reasons to go with mDNS in bootstrap were a) Bootstrap should be reusable also outside of full anima solutions and introduce as few as possible new elements. mDNS was considered well-known and very-likely even pre-existing in low-end devices (IoT). b) Using GRASP both for the insecure neighbor discovery and secure ACP operations raises the risk of introducing security issues through implementation issues/non-isolation between those two instances of GRASP.
- Shortened the section on GRASP instances, because with mDNS being used for discovery, there is no insecure GRASP session any longer, simplifying the GRASP considerations.
- Added certificate requirements for ANIMA in section 5.1.1, specifically how the ANIMA information is encoded in subjectAltName.
- Deleted the appendix on "ACP without separation", as originally planned, and the paragraph in the main text referring to it.
- Deleted one sub-addressing scheme, focusing on a single scheme now.
- Included information on how ANIMA information must be encoded in the domain certificate in section "preconditions".
- Editorial changes, updated draft references, etc.

14.10. draft-ietf-anima-autonomic-control-plane-04

Changed discovery of ACP neighbor back from mDNS to GRASP after revisiting the L2 problem. Described problem in discovery section itself to justify. Added text to explain how ACP discovery relates to BRSKY (bootstrap) discovery and pointed to Michael Richardsons draft detailing it. Removed appendix section that contained the original explanations why GRASP would be useful (current text is meant to be better).
14.11. draft-ietf-anima-autonomic-control-plane-05

- Section 5.3 (candidate ACP neighbor selection): Add that Intent can override only AFTER an initial default ACP establishment.

- Section 6.10.1 (addressing): State that addresses in the ACP are permanent, and do not support temporary addresses as defined in RFC4941.

- Modified Section 6.3 to point to the GRASP objective defined in draft-carpenter-anima-ani-objectives. (and added that reference)

- Section 6.10.2: changed from MD5 for calculating the first 40-bits to SHA256; reason is MD5 should not be used any more.

- Added address sub-scheme to the IANA section.

- Made the routing section more prescriptive.

- Clarified in Section 8.1.1 the ACP Connect port, and defined that term "ACP Connect".

- Section 8.2: Added some thoughts (from mcr) on how traversing a L3 cloud could be automated.

- Added a CRL check in Section 6.7.

- Added a note on the possibility of source-address spoofing into the security considerations section.

- Other editorial changes, including those proposed by Michael Richardson on 30 Nov 2016 (see ANIMA list).

14.12. draft-ietf-anima-autonomic-control-plane-06

- Added proposed RPL profile.

- detailed DTLS profile - DTLS with any additional negotiation/signaling channel.

- Fixed up text for ACP/GRE encap. Removed text claiming its incompatible with non-GRE IPsec and detailed it.

- Added text to suggest admin down interfaces should still run ACP.
14.13. draft-ietf-anima-autonomic-control-plane-07

- Changed author association.

- Improved ACP connect section (after confusion about term came up in the stable connectivity draft review). Added picture, defined complete terminology.

- Moved ACP channel negotiation from normative section to appendix because it can in the timeline of this document not be fully specified to be implementable. Aka: work for future document. That work would also need to include analysing IKEv2 and describing the difference of a proposed GRASP/TLS solution to it.

- Removed IANA request to allocate registry for GRASP/TLS. This would come with future draft (see above).

- Gave the name "ACP domain information field" to the field in the certificate carrying the ACP address and domain name.

- Changed the rules for mutual authentication of certificates to rely on the domain in the ACP information field of the certificate instead of the OU in the certificate. Also renewed the text pointing out that the ACP information field in the certificate is meant to be in a form that it does not disturb other uses of the certificate. As long as the ACP expected to rely on a common OU across all certificates in a domain, this was not really true: Other uses of the certificates might require different OUs for different areas/type of devices. With the rules in this draft version, the ACP authentication does not rely on any other fields in the certificate.

- Added an extension field to the ACP information field so that in the future additional fields like a subdomain could be inserted. An example using such a subdomain field was added to the pre-existing text suggesting sub-domains. This approach is necessary so that there can be a single (main) domain in the ACP information field, because that is used for mutual authentication of the certificate. Also clarified that only the register(s) SHOULD/MUST use that the ACP address was generated from the domain name - so that we can easier extend change this in extensions.

- Took the text for the GRASP discovery of ACP neighbors from Brian's grasp-ani-objectives draft. Alas, that draft was behind the latest GRASP draft, so i had to overhaul. The mayor change is to describe in the ACP draft the whole format of the M_FLOOD message (and not only the actual objective). This should make it a lot easier to read (without having to go back and forth to the GRASP
RFC/draft). It was also necessary because the locator in the M_FLOOD messages has an important role and its not coded inside the objective. The specification of how to format the M_FLOOD message should now be complete, the text may be some duplicate with the DULL specificaion in GRASP, but no contradiction.

- One of the main outcomes of reworking the GRASP section was the notion that GRASP announces both the candidate peers IPv6 link local address but also the support ACP security protocol including the port it is running on. In the past we shied away from using this information because it is not secured, but I think the additional attack vectors possible by using this information are negligible: If an attacker on an L2 subnet can fake another devices GRASP message then it can already provide a similar amount of attack by purely faking the link-local address.

- Removed the section on discovery and BRSKI. This can be revived in the BRSKI document, but it seems mood given how we did remove mDNS from the latest BRSKI document (aka: this section discussed discrepancies between GRASP and mDNS discovery which should not exist anymore with latest BRSKI.

- Tried to resolve the EDNOTE about CRL vs. OCSP by pointing out we do not specify which one is to be used but that the ACP should be used to reach the URL included in the certificate to get to the CRL storage or OCSP server.

- Changed ACP via IPsec to ACP via IKEv2 and restructured the sections to make IPsec native and IPsec via GRE subsections.

- No need for any assigned DTLS port if ACP is run across DTLS because it is signaled via GRASP.

14.14. draft-ietf-anima-autonomic-control-plane-08

Modified mentioning of BRSKI to make it consistent with current (07/2017) target for BRSKI: MASA and IDevID are mandatory. Devices with only insecure UDI would need a security reduced variant of BRSKI. Also added mentioning of Netconf Zero-Touch. Made BRSKI non-normative for ACP because wrt. ACP it is just one option how the domain certificate can be provisioned. Instead, BRSKI is mandatory when a device implements ANI which is ACP+BRSKI.

Enhanced text for ACP across tunnels to describe two options: one across configured tunnels (GRE, IPinIP etc) a more efficient one via directed DULL.
Moved description of BRSKI to appendix to emphasize that BRSKI is not a (normative) dependency of GRASP, enhanced text to indicate other options how Domain Certificates can be provisioned.

Added terminology section.

Separated references into normative and non-normative.

Enhanced section about ACP via "tunnels". Defined an option to run ACP secure channel without an outer tunnel, discussed PMTU, benefits of tunneling, potential of using this with BRSKI, made ACP via GREP a SHOULD requirement.

Moved appendix sections up before IANA section because there where concerns about appendices to be too far on the bottom to be read. Added (Informative) / (Normative) to section titles to clarify which sections are informative and which are normative.

Moved explanation of ACP with L2 from precondition to separate section before workarounds, made it instructive enough to explain how to implement ACP on L2 ports for L3/L2 switches and made this part of normative requirement (L2/L3 switches SHOULD support this).

Rewrote section "GRASP in the ACP" to define GRASP in ACP as mandatory (and why), and define the ACP as security and transport substrate to GRASP in ACP. And how it works.

Enhanced "self-protection" properties section: protect legacy management protocols. Security in ACP is for protection from outside and those legacy protocols. Otherwise need end-to-end encryption also inside ACP, e.g., with domain certificate.

Enhanced initial domain certificate section to include requirements for maintenance (renewal/revocation) of certificates. Added explanation to BRSKI informative section how to handle very short lived certificates (renewal via BRSKI with expired cert).

Modified the encoding of the ACP address to better fit RFC822 simple local-parts (":" as required by RFC5952 are not permitted in simple dot-atoms according to RFC5322. Removed reference to RFC5952 as its now not needed anymore.

Introduced a sub-domain field in the ACP information in the certificate to allow defining such subdomains with depending on future Intent definitions. It also makes it clear what the "main domain" is. Scheme is called "routing subdomain" to have a unique name.
Added V8 (now called Vlong) addressing sub-scheme according to suggestion from mcr in his mail from 30 Nov 2016 (https://mailarchive.ietf.org/arch/msg/anima/n2pEPhrTqDCBdzsKMPaIn2gsIzI). Also modified the explanation of the single V bit in the first sub-scheme now renamed to Zone sub-scheme to distinguish it.

14.15. draft-ietf-anima-autonomic-control-plane-09

Added reference to RFC4191 and explained how it should be used on ACP edge routers to allow auto configuration of routing by NMS hosts. This came after review of stable connectivity draft where ACP connect is being referred to.

V8 addressing Sub-Scheme was modified to allow not only /8 device-local address space but also /16. This was in response to the possible need to have maybe as much as 2^12 local addresses for future encaps in BRISKI like IPinIP. It also would allow fully autonomic address assignment for ACP connect interfaces from this local address space (on an ACP edge device), subject to approval of the implied update to rfc4291/rfc4193 (IID length). Changed name to Vlong addressing sub-scheme.


- The stable connectivity draft was vaguely describing ACP connect behavior that is better standardized in this ACP draft.
- Added new ACP "Manual" addressing sub-scheme with /64 subnets for use with ACP connect interfaces. Being covered by the ACP ULA prefix, these subnets do not require additional routing entries for NMS hosts. They also are fully 64-bit IID length compliant and therefore not subject to 4191bis considerations. And they avoid that operators manually assign prefixes from the ACP ULA prefixes that might later be assigned autonomically.
- ACP connect auto-configuration: Defined that ACP edge devices, NMS hosts should use RFC4191 to automatically learn ACP prefixes. This is especially necessary when the ACP uses multiple ULA prefixes (via e.g., the rsub domain certificate option), or if ACP connect sub-interfaces use manually configured prefixes NOT covered by the ACP ULA prefixes.
- Explained how rfc6724 is (only) sufficient when the NMS host has a separate ACP connect and Data-Plane interface. But not when there is a single interface.
Added a separate subsection to talk about "software" instead of "NMS hosts" connecting to the ACP via the "ACP connect" method. The reason is to point out that the "ACP connect" method is not only a workaround (for NMS hosts), but an actual desirable long term architectural component to modularly build software (e.g., ASA or OAM for VNF) into ACP devices.

Added a section to define how to run ACP connect across the same interface as the Data-Plane. This turns out to be quite challenging because we only want to rely on existing standards for the network stack in the NMS host/software and only define what features the ACP edge device needs.

Added section about use of GRASP over ACP connect.

Added text to indicate packet processing/filtering for security: filter incorrect packets arriving on ACP connect interfaces, diagnose on RPL root packets to incorrect destination address (not in ACP connect section, but because of it).

Reaffirm security goal of ACP: Do not permit non-ACP routers into ACP routing domain.

Made this ACP document be an update to RFC4291 and RFC4193. At the core, some of the ACP addressing sub-schemes do effectively not use 64-bit IIDs as required by RFC4191 and debated in rfc4191bis. During 6man in Prague, it was suggested that all documents that do not do this should be classified as such updates. Add a rather long section that summarizes the relevant parts of ACP addressing and usage and. Aka: This section is meant to be the primary review section for readers interested in these changes (e.g., 6man WG.).

Added changes from Michael Richarsons review https://github.com/anima-wg/autonomic-control-plane/pull/3/commits, textual and:

ACP discovery inside ACP is bad *doh*!.

Better CA trust and revocation sentences.

More details about RPL behavior in ACP.

Black hole route to avoid loops in RPL.

Added requirement to terminate ACP channels upon cert expiry/revocation.

Added fixes from 08-mcr-review-reply.txt (on github):
- AN Domain Names are FQDNs.
- Fixed bit length of schemes, numerical writing of bits (00b/01b).

### 14.16. draft-ietf-anima-autonomic-control-plane-10

Used the term routing subdomain more consistently where previously only subdomain was used. Clarified use of routing subdomain in creation of ULA "global ID" addressing prefix.

6.7.1.* Changed native IPsec encapsulation to tunnel mode (necessary), explained why. Added notion that ESP is used, added explanations why tunnel/transport mode in native vs. GRE cases.

6.10.3/6.10.5 Added term "ACP address range/set" to be able to better explain how the address in the ACP certificate is actually the base address (lowest address) of a range/set that is available to the device.

6.10.4 Added note that manual address sub-scheme addresses must not be used within domain certificates (only for explicit configuration).

6.12.5 Refined explanation of how ACP virtual interfaces work (p2p and multipoint). Did seek for pre-existing RFCs that explain how to build a multi-access interface on top of a full mesh of p2p connections (6man WG, anima WG mailing lists), but could not find any prior work that had a succinct explanation. So wrote up an explanation here. Added hopefully all necessary and sufficient details how to map ACP unicast packets to ACP secure channel, how to deal with ND packet details. Added verbiage for ACP not to assign the virtual interface link-local address from the underlying interface. Added note that GRAP link-local messages are treated specially but logically the same. Added paragraph about NBMA interfaces.

remaining changes from Brian Carpenters review. See Github file draft-ietf-anima-autonomic-control-plane/08-carpenter-review-reply.txt for more details:

- Added multiple new RFC references for terms/technologies used.
- Fixed verbiage in several places.

2. (terminology) Added 802.1AR as reference.

2. Fixed up definition of ULA.
6.1.1 Changed definition of ACP information in cert into ABNF format. Added warning about maximum size of ACP address field due to domain-name limitations.

6.2 Mentioned API requirement between ACP and clients leveraging adjacency table.

6.3 Fixed TTL in GRASP example: msec, not hop-count!.

6.8.2 MAYOR: expanded security/transport substrate text:

Introduced term ACP GRASP virtual interface to explain how GRASP link-local multicast messages are encapsulated and replicated to neighbors. Explain how ACP knows when to use TLS vs. TCP (TCP only for link-local address (sockets)). Introduced "ladder" picture to visualize stack.

6.8.2.1 Expanded discussion/explanation of security model. TLS for GRASP unicast connections across ACP is double encryption (plus underlying ACP secure channel), but highly necessary to avoid very simple man-in-the-middle attacks by compromised ACP members on-path. Ultimately, this is done to ensure that any apps using GRASP can get full end-to-end secrecy for information sent across GRASP. But for publically known ASA services, even this will not provide 100% security (this is discussed). Also why double encryption is the better/easier solution than trying to optimize this.

6.10.1 Added discussion about pseudo-random addressing, scanning-attacks (not an issue for ACP).

6.12.2 New performance requirements section added.

6.10.1 Added notion to first experiment with existing addressing schemes before defining new ones – we should be flexible enough.

6.3/7.2 clarified the interactions between MLD and DULL GRASP and specified what needs to be done (e.g., in 2 switches doing ACP per L2 port).

12. Added explanations and cross-references to various security aspects of ACP discussed elsewhere in the document.

13. Added IANA requirements.

Added RFC2119 boilerplate.
14.17. draft-ietf-anima-autonomic-control-plane-11

Same text as -10 Unfortunately when uploading -10 .xml/.txt to datatracker, a wrong version of .txt got uploaded, only the .xml was correct. This impacts the -10 html version on datatracker and the PDF versions as well. Because rfcdiff also compares the .txt version, this -11 version was created so that one can compare changes from -09 and changes to the next version (-12).

14.18. draft-ietf-anima-autonomic-control-plane-12

Sheng Jiangs extensive review. Thanks! See Github file draft-ietf-anima-autonomic-control-plane/09-sheng-review-reply.txt for more details. Many of the larger changes listed below where inspired by the review.

Removed the claim that the document is updating RFC4291,RFC4193 and the section detailing it. Done on suggestion of Michael Richardson - just try to describe use of addressing in a way that would not suggest a need claim update to architecture.

Terminology cleanup:

- Replaced "device" with "node" in text. Kept "device" only when referring to "physical node". Added definitions for those words. Includes changes of derived terms, especially in addressing: "Node-ID" and "Node-Number" in the addressing details.

- Replaced term "autonomic FOOBAR" with "acp FOOBAR" as wherever appropriate: "autonomic" would imply that the node would need to support more than the ACP, but that is not correct in most of the cases. Wanted to make sure that implementers know they only need to support/implement ACP - unless stated otherwise. Includes "AN->ACP node", "AN->ACP adjacency table" and so on.

1 Added explanation in the introduction about relationship between ACP, BRSKI, ANI and Autonomic Networks.

6.1.1 Improved terminology and features of the certificate information field. Now called domain information field instead of ACP information field. The acp-address field in the domain information field is now optional, enabling easier introduction of various future options.

6.1.2 Moved ACP domain membership check from section 6.6 to (ACP secure channels setup) here because it is not only used for ACP secure channel setup.
6.1.3 Fix text about certificate renewal after discussion with Max Pritikin/Michael Richardson/Brian Carpenter:

- Version 10 erroneously assumed that the certificate itself could store a URL for renewal, but that is only possible for CRL URLs. Text now only refers to "remembered EST server" without implying that this is stored in the certificate.

- Objective for RFC7030/EST domain certificate renewal was changed to "SRV.est" See also IANA section for explanation.

- Removed detail of distance based service selection. This can be better done in future work because it would require a lot more detail for a good DNS-SD compatible approach.

- Removed detail about trying to create more security by using ACP address from certificate of peer. After rethinking, this does not seem to buy additional security.

6.10 Added reference to 6.12.5 in initial use of "loopback interface" in section 6.10 in result of email discussion michaelR/michaelB.

10.2 Introduced informational section (diagnostics) because of operational experience - ACP/ANI undeployable without at least diagnostics like this.

10.3 Introduced informational section (enabling/disabling) ACP. Important to discuss this for security reasons (e.g., why to never auto-enable ANI on brownfield devices), for implementers and to answer ongoing questions during WG meetings about how to deal with shutdown interface.

10.8 Added informational section discussing possible future variations of the ACP for potential adopters that cannot directly use the complete solution described in this document unmodified.

14.19. draft-ietf-anima-autonomic-control-plane-13

Swap author list (with permission).

6.1.1. Eliminate blank lines in definition by making it a picture (reformatting only).

6.10.3.1 New paragraph: Explained how nodes using Zone-ID != 0 need to use Zone-ID != 0 in GRASP so that we can avoid routing/forwarding of Zone-ID = 0 prefixes.
Rest of feedback from review of -12, see
https://raw.githubusercontent.com/anima-wg/autonomic-control-
plane/master/draft-ietf-anima-autonomic-control-plane/12-feedback-
reply.txt

Review from Brian Carpenter:

various: Autonomous -> autonomic(ally) in all remaining occurrences.

various: changed "manual (configured)" to "explicitly (configured)"
to not exclude the option of (SDN controller) automatic configuration
(no humans involved).


2. Added definition of loopback interface == internal interface.
   After discus on WG mailing lists, including 6man.

6.1.2 Defined CDP/OCSP and pointed to RFC5280 for them.

6.1.3 Removed "EST-TLS", no objective value needed or beneficial,
   added explanation paragraph why.

6.2 Added to adjacency table the interface that a neighbor is
   discovered on.

6.3 Simplified CDDL syntax: Only one method per AN_ACP objective
   (because of locators). Example with two objectives in GRASP message.

6.8.1 Added note about link-local GRASP multicast message to avoid
   confusion.

8.1.4 Added RFC8028 as recommended on hosts to better support VRF-
   select with ACP.

8.2.1 Rewrote and Simplified CDDL for configured remote peer and
   explanations. Removed pattern option for remote peer. Not important
   enough to be mandated.

Review thread started by William Atwood:

2. Refined definition of VRF (vs. MPLS/VPN, LISP, VRF-LITE).

2. Refined definition of ACP (ACP includes ACP GRASP instance).

2. Added explanation for "zones" to terminology section and into
   Zone Addressing Sub Scheme section, relating it to RFC4007 zones
   (from Brian Carpenter).
4. Fixed text for ACP4 requirement (Clients of the ACP must not be tied to specific protocol.).

5. Fixed step 4. with proposed text.

6.1.1 Included suggested explanation for rsub semantics.

6.1.3 must->MUST for at least one EST server in ACP network to autonomically renew certs.

6.7.2 normative: AND MUST NOT (permit weaker crypto options.

6.7.1.1 also included text denying weaker IPsec profile options.

6.8.2 Fixed description how to build ACP GRASP virtual interfaces. Added text that ACP continues to exist in absence of ACP neighbors.

various: Make sure all "zone" words are used consistently.

6.10.2/various: fixed 40-bit RFC4193 ULA prefix in all examples to 89b714f3db (thanks MichaelR).

6.10.1 Removed comment about assigned ULA addressing. Decision not to use it now ancient history of WG decision making process, not worth nothing anymore in the RFC.

Review from Yongkang Zhang:

6.10.5 Fixed length of Node-Numbers in ACP Vlong Addressing Sub-Scheme.

14.20. draft-ietf-anima-autonomic-control-plane-14

Disclaimer: All new text introduced by this revision provides only additional explanations/ details based on received reviews and analysis by the authors. No changes to behavior already specified in prior revisions.

Joel Halpern, review part 3:

Define/explain "ACP registrar" in reply to Joel Halpern review part 3, resolving primarily 2 documentation issues::

1. Unclear how much ACP depends on BRSKI. ACP document was referring unqualified to registrars and Registrar-ID in the addressing section without explaining what a registrar is, leading to the assumption it must be a BRSKI Registrar.
2. Unclear how the ACP addresses in ACP domain certificates are assigned because the BRSKI document does not defines this, but refers to this ACP document.

Wrt. 1: ACP does NOT depend on BRSKI registrars, instead ANY appropriate automated or manual mechanism can be used to enroll ACP nodes with ACP domain certificates. This revision calls defines such mechanisms the "ACP registrar" and defines requirements. this is non-normative, because it does not define specific mechanisms that need to be support. In ANI devices, ACP Registrars are BRSKI Registrars. In non-ANI ACP networks, the registrar may simply be a person using CLI/web-interfaces to provision domain certificates and set the ACP address correctly in the ACP domain certificate.

Wrt. 2.: The BRSKI document does rightfully not define how the ACP address assignment and creation of the ACP domain information field has to work because this is independent of BRSKI and needs to follow the same rules whatever protocol/mechanisms are used to implement an ACP Registrar. Another set of protocols that could be used instead of BRSKI is Netconf/Netconf-Call-Home, but such an alternative ACP Registrar solution would need to be specified in its own document.

Additional text/sections had to be added to detail important conditions so that automatic certificate maintenance for ACP nodes (with BRSKI or other mechanisms) can be done in a way that as good as possible maintains ACP address information of ACP nodes across the nodes lifetime because that ACP address is intended as an identifier of the ACP node.

Summary of sections added:

- 6.1.3.5/6.1.3.6 (normative): re-enrollment of ACP nodes after certificate expiry/failure in a way that allows to maintain as much as possible ACP address information.
- 6.10.7 (normative): defines "ACP Registrar" including requirements and how it can perform ACP address assignment.
- 10.3 (informative): details / examples about registrars to help implementers and operators understand easier how they operate, and provide suggestion of models that a likely very useful (sub-CA and/or centralized policy management).
- 10.4 (informative): Explains the need for the multiple address sub-spaces defined in response to discuss with Joel.

Other changes:
Updated references (RFC8366, RFC8368).

Introduced sub-section headings for 6.1.3 (certificate maintenance) because section became too long with newly added sub-sections. Also some small text fixups/remove of duplicate text.

Gen-ART review, Elwyn Davies:

[RFC Editor: how can i raise the issue of problematic cross references of terms in the terminology section - rendering is problematic. ].

4. added explanation for ACP4 (finally).

6.1.1 Simplified text in bullet list explaining rfc822 encoding.

6.1.3 refined second paragraph defining remembering of previous EST server and explaining how to do this with BRSKI.

9.1 Added paragraph outlining the benefit of the sub-CA Registrar option for supporting partitioned networks.

Roughly 100 more nits/minor fixes throughout the document. See: https://raw.githubusercontent.com/anima-wg/autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/13-elwynd-reply.txt

Joel Halpern, review part 2:

6.1.1: added note about "+ +" format in address field when acp-address and rsub are empty.

6.5.10 - clarified text about V bit in Vlong addressing scheme.

6.10.3/6.10.4 - moved the Z bit field up front (directly after base scheme) and indicated more explicitly Z is part of selecting of the sub-addressing scheme.

Refined text about reaching CRL Distribution Point, explain why address as indicator to use ACP.

Note from Brian Carpenter: RFC Editor note for section reference into GRASP.

IOT directorate review from Pascal Thubert:

Various Nits/typos.
TBD: Punted wish for mentioning RFC reference titles to RFC editor for now.

1. Added section 1.1 - applicability, discussing protocol choices re. applicability to constrained devices (or not). Added notion of TCP/TLS via CoAP/DTLS to section 10.4 in support of this.


5. Referenced section 8.2 for remote ACP channel configuration.

6.3 made M_FLOOD periods RECOMMENDED (less guesswork)

6.7.x Clarified conditional nature of MUST for the profile details of IPsec parameters (aka: only 6.7.3 defines actual MUST for nodes, prior notions only define the requirements for IPsec profiles IF IPsec is supported.)

6.8.1 Moved discussion about IP multicast, IGP, RPL for GRASP into a new subsection in the informative part (section 10) to tighten up text in normative part.

6.10.1 added another reference to stable-connectivity for interop with IPv4 management.

6.10.1 removed mentioning of ULA-Random, term was used in email discus of ULA with L=1, but term actually not defined in rfc4193, so mentioning it is just confusing/redundant. Also added note about the random hash being defined in this document, not using SHA1 from rfc4193.

6.11.1.1 added suggested text about mechanisms to further reduce opportunities for loop during reconvergence (active signaling options from RFC6550).

6.11.1.3 made mode 2 MUST and mode 2 MAY (RPL MOP - mode of operations). Removes ambiguity.

6.12.5 Added recommendation for RFC4429 (optimistic DAD).

Nits from Benjamin Kaduk: dTLS -> DTLS:

Review from Joel Halpern:

1. swapped order of "purposes" for ACP to match order in section 3.

1. Added notion about manageability of ACP gong beyond RFC7575 (before discussion of stable connectivity).
2. Changed definition of Intent to be same as reference model (policy language instead of API).

6.1.1 changed BNF specification so that a local-part without acp-address (for future extensions) would not be rfcSELF+rsbut but simpler rfcSELF+rsb. Added explanation why rsb is in local-part.

Tried to eliminate unnecessary references to VRF to minimize assumption how system is designed.

6.1.3 Explained how to make CDP reachable via ACP.

6.7.2 Made it clearer that constrained devices MUST support DTLS if they cannot support IPsec.

6.8.2.1 clarified first paragraph (TCP retransmissions lightweight).

6.11.1 fixed up RPL profile text - to remove "VRF". Text was also buggy. mentioned control plane, but it’s a forwarding/silicon issue to have these header.

6.12.5 Clarified how link-local ACP channel address can be derived, and how not.

8.2.1 Fixed up text to distinguish between configuration and model describing parameters of the configuration (spec only provides parameter model).

Various Nits.

14.21. draft-ietf-anima-autonomic-control-plane-15

Only reshuffling and formatting changes, but wanted to allow reviewers later to easily compare -13 with -14, and these changes in -15 mess that up too much.

increased TOC depth to 4.

Separated and reordered section 10 into an operational and a background and futures section. The background and futures could also become appendices if the layout of appendices in RFC format wasn’t so horrible that you really only want to avoid using them (all the way after a lot of text like references that stop most readers from proceeding any further).
14.22. draft-ietf-anima-autonomic-control-plane-16

Mirja Kuehlewind:

Tightened requirements for ACP related GRASP objective timers.

Better text to introduce/explains baseline and constrained ACP profiles.

IANA guideline: MUST only accept extensible last allocation for address sub-scheme.

Moved section 11 into appendix.

Warren Kumari:

Removed "global routing table", replaced with "Data-Plane routing (and forwarding) tables.

added text to indicate how routing protocols do like to have data-plane dependencies.

Changed power consumption section re. admin-down state. Power needed to bring up such interfaces make t inappropriate to probe. Need to think more about best suggests -> beyond scope.

Replaced "console" with out-of-band... (console/management ethernet).

Various nits.

Joel Halpern:

Fixed up domain information field ABNF to eliminate confusion that rsub is not an FQDN but only a prefix to routing-subdomain.

Corrected certcheck to separate out cert verification into lifetime validity and proof of ownership of private key.

Fixed pagination for "ACP as security and transport substrate for GRASP" picture.

14.23. draft-ietf-anima-autonomic-control-plane-17

Review Alissa Cooper:

Main discuss point fixed by untangling two specific node type cases:
NOC nodes have ACP domain cert without acp-address field. Are ACP domain members, but cannot build ACP secure channels (just end-to-end or any other authentications.

ACP nodes may have other methods to assign ACP address than getting it through the cert. This is indicated through new value 0 for acp-address in certificate.

Accordingly modified texts in ABNF/explanation and Cert-Check section.

Other:

Better separation of normative text and considerations for "future" work:

- Marked missing chapters as Informative. Reworded requirements section to indicate its informative nature, changed requirements to _MUST_/ _SHOULD_ to indicate these are not RFC2119 requirements but that this requirements section is really just in place of a separate solutions requirements document (that ANIMA was not allowed to produce).

- removed ca. 20 instances of "futures" in normative part of document.

- moved important instances of "futures" into new section A.10 (last section of appendix). These serve as reminder of work discussed during WG but not able to finish specifying it.

Eliminated perception that "rsub" (routing subdomain) is only beneficial with future work. Example in A.7.

Added RFC-editor note re formatting of references to terms defined in terminology section.

Using now correct RFC 8174 boilerplate.

Clarified semantic and use of manual ACP sub-scheme. Not used in certificates, only assigned via traditional methods. Use for ACP-connect subnets or the like.

Corrected text about Data-Plane dependencies of ACP. Appropriate implementations can be fully data-plane independent (without more spec work) if not sharing link-local address with Data-Plane. 6.12.2 text updated to discuss those (MAC address), A.10.2 discusses options that would require new standards work.
Moved all text about Intent into A.8 to clearly mark it as futures.

Changed suggestion of future insecure ACP option to future "end-to-end-security-only" option.

Various textual fixes.

Gen-ART review by Elwyn Davies:

Some fixes also mentioned by Alissa.

Added reference for OT.

Fixed notion that secure channel is not only a security association.

>20 good textual fixes. Thanks!

Other:

Added picture requested by Pascal Thubert about Dual-NOC (A.10.4).

Moved RFC-editor request for better first RFC reference closer to the top of the document.

Fixed typo /126 -> 127 for prefix length with zone address scheme.

Overlooked early SecDir review from frank.xialiang@huawei.com:

most issues fixed through other review in -16. Added reference to self-protection section 9.2 into security considerations section.

14.24. draft-ietf-anima-autonomic-control-plane-18

Too many word/grammar mistakes in -17.

14.25. draft-ietf-anima-autonomic-control-plane-19

Review Eric Rescola:

6.1.2 - clarified that we do certificate path validation against potentially multiple trust anchors.

6.1.3 - Added more comprehensive explanation of Trust Points via new section 6.1.3.

6.5 - added figure with sequential steps of ACP channel establishment and Alice and Bob finding their role in the setup.
6.7.x - detailed crypto profiles: AES-256-GCM, ECDHE.

6.7.2 - Referring to RFC7525 as the required crypto profile for DTLS (taking text from RFC8310 as previously discussed with Eric).

6.7.3 - Added explanation that ACP needs no single MTI secure channel protocol with example.

6.10.2 - Added requirement that rsub must be chosen so that they don't create SHA256 collisions. Added explanation how the same could be done for different ACP networks with same trust anchors but that this outside the scope of this specification.

6.7.10 - Explains security expectations against ACP registrars: Must be trusted and then given credentials to act as FKI RA to help pledges to enroll with an ACP certificate.

9.1 - Added explanations about merging ACP domains requiring both domains to trust union of Trust Anchors and need to avoid ULA hash collisions.

11 - Added that ACP registrars are critical infrastructure requiring hardening like CA, mentioning attack impact examples.

11 - Mentioning that ACP requires initial setup of CA and registrar.

11 - Long rewrite/extension of group security model and its implication shared with review from Ben (below).

Many nits fixed.

Review Benjamin Kaduk:

Fixed various nits.

Changed style of MUST/SHOULD in Requirements section to all lower case to avoid any RFC2119 confusion.

1. clarified support for constrained devices/DTLS: Opportunistic.

1. Clarified ACPs use of two variants of GRASP DULL for neighbor discovery and ACP grasp for service discovery/clients.

3.2 - amended text explaining what additional security ACP provides for bootstrap protocols.

6.1.1 - Added note about ASN.1 encoding in the justification for use of rfc822address.
6.1.2 - Added details how to handle ACP connection when node via which OCSP/CRL-server is reached fails certificate verification.

12. Rewrote explanation why objective names requested for ACP use SRV.name.

10.4 - added summary section about ACP and configuration.

Review Eric Rescorla:

6.1.2 - changed peer certificate verification to be certificate path verification, added lowercase normalization comparison to domain name check.

6.1.2 - explained how domain membership check is authentication and authorization.

6.1.4.1 - Fixed "objective value" to "objective name".

6.1.4.3 - check IPv6 address of CDP against CDP ACP certificate IPv6 address only if URL uses IPv6 address.

6.10.1 - added more justification why there is no need for privacy protection of ACP addresses.

6.11.1.1 - thorough fixup of sentences/structure of this RPL overview section to make it more logical and easier to digest. Also added a paragraph about the second key benefit of this profile (scalability).

6.11.1.9 - Added explanation about not using RPL security from Benjamin.

8.1.1 - Fixed up text for address assignment of ACP connect interfaces. Only recommending manual addressing scheme.

9.1 - changed self-healing benefit text to describe immediate channel reset for short-lived certificates and describing how the same with CRL/OCSP is optional.

11. - added note about immediate termination of secure channels after certificate expiry as this is uncommon today.

11. - rewrote section of security model, attacks and mitigation of compromised ACP members.

A.24 - clarified the process in which expired certificates are used for certificate renewal to avoid higher overhead of re-enrolment.
A.4 - removed mentioning of RPL trickle because not used by ACP RPL profile.

A.10.8 - added section discussing how to minimize risk of compromised nodes, recovering them or kicking them out.


Need to find good reference for TLS profile for ACP GRASP TLS connections.

TBD: Add DTLS choice to GRASP secure channel.

15. References

15.1. Normative References

[I-D.ietf-anima-grasp]

[I-D.ietf-cbor-cddl]


15.2. Informative References


Appendix A. Background and Futures (Informative)

The following sections discuss additional background information about aspects of the normative parts of this document or associated mechanisms such as BRSKI (such as why specific choices were made by the ACP) and they provide discussion about possible future variations of the ACP.

A.1. ACP Address Space Schemes

This document defines the Zone, Vlong and Manual sub address schemes primarily to support address prefix assignment via distributed, potentially uncoordinated ACP registrars as defined in Section 6.10.7. This costs 48/46-bit identifier so that these ACP registrar can assign non-conflicting address prefixes. This design does not leave enough bits to simultaneously support a large number of nodes (Node-ID) plus a large prefix of local addresses for every node plus a large enough set of bits to identify a routing Zone. In result, Zone, Vlong 8/16 attempt to support all features, but in via separate prefixes.

In networks that always expect to rely on a centralized PMS as described above (Section 10.2.5), the 48/46-bits for the Registrar-ID could be saved. Such variations of the ACP addressing mechanisms could be introduced through future work in different ways. If the prefix rfcSELF in the ACP information field was changed, incompatible ACP variations could be created where every design aspect of the ACP could be changed. Including all addressing choices. If instead a new addressing sub-type would be defined, it could be a backward compatible extension of this ACP specification. Information such as
the size of a zone-prefix and the length of the prefix assigned to
the ACP node itself could be encoded via the extension field of the
ACP domain information.

Note that an explicitly defined "Manual" addressing sub-scheme is
always beneficial to provide an easy way for ACP nodes to prohibit
incorrect manual configuration of any non-"Manual" ACP address spaces
and therefore ensure that "Manual" operations will never impact
correct routing for any non-"Manual" ACP addresses assigned via ACP
domain certificates.

A.2. BRSKI Bootstrap (ANI)

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) describes how nodes
with an IDevID certificate can securely and zero-touch enroll with a
domain certificate (LDevID) to support the ACP. BRSKI also leverages
the ACP to enable zero-touch bootstrap of new nodes across networks
without any configuration requirements across the transit nodes
(e.g., no DHCP/DNS forwarding/server setup). This includes otherwise
not configured networks as described in Section 3.2. Therefore BRSKI
in conjunction with ACP provides for a secure and zero-touch
management solution for complete networks. Nodes supporting such an
infrastructure (BRSKI and ACP) are called ANI nodes (Autonomic
Networking Infrastructure), see [I-D.ietf-anima-reference-model].
Nodes that do not support an IDevID but only an (insecure) vendor
specific Unique Device Identifier (UDI) or nodes whose manufacturer
does not support a MASA could use some future security reduced
version of BRSKI.

When BRSKI is used to provision a domain certificate (which is called
enrollment), the BRSKI registrar (acting as an enhanced EST server)
must include the subjectAltName / rfc822Name encoded ACP address and
domain name to the enrolling node (called pledge) via its response to
the pledges EST CSR Attribute request that is mandatory in BRSKI.

The Certificate Authority in an ACP network must not change the
subjectAltName / rfc822Name in the certificate. The ACP nodes can
therefore find their ACP address and domain using this field in the
domain certificate, both for themselves, as well as for other nodes.

The use of BRSKI in conjunction with the ACP can also help to further
simplify maintenance and renewal of domain certificates. Instead of
relying on CRL, the lifetime of certificates can be made extremely
small, for example in the order of hours. When a node fails to
connect to the ACP within its certificate lifetime, it cannot connect
to the ACP to renew its certificate across it (using just EST), but
it can still renew its certificate as an "enrolled/expired pledge"
via the BRSKI bootstrap proxy. This requires only that the BRSKI
registrar honors expired domain certificates and that the pledge attempts to perform TLS authentication for BRSKI bootstrap using its expired domain certificate before falling back to attempting to use its IDevID for BRSKI. This mechanism could also render CRLs unnecessary because the BRSKI registrar in conjunction with the CA would not renew revoked certificates - only a "Do-not-renew" list would be necessary on BRSKI registrars/CA.

In the absence of BRSKI or less secure variants thereof, provisioning of certificates may involve one or more touches or non-standardized automation. Node vendors usually support provisioning of certificates into nodes via PKCS#7 (see [RFC2315]) and may support this provisioning through vendor specific models via Netconf ([RFC6241]). If such nodes also support Netconf Zero-Touch ([I-D.ietf-netconf-zerotouch]) then this can be combined to zero-touch provisioning of domain certificates into nodes. Unless there are equivalent integration of Netconf connections across the ACP as there is in BRSKI, this combination would not support zero-touch bootstrap across a not configured network though.

A.3. ACP Neighbor discovery protocol selection

This section discusses why GRASP DULL was chosen as the discovery protocol for L2 adjacent candidate ACP neighbors. The contenders considered where GRASP, mDNS or LLDP.

A.3.1. LLDP

LLDP and Cisco’s earlier Cisco Discovery Protocol (CDP) are example of L2 discovery protocols that terminate their messages on L2 ports. If those protocols would be chosen for ACP neighbor discovery, ACP neighbor discovery would therefore also terminate on L2 ports. This would prevent ACP construction over non-ACP capable but LLDP or CDP enabled L2 switches. LLDP has extensions using different MAC addresses and this could have been an option for ACP discovery as well, but the additional required IEEE standardization and definition of a profile for such a modified instance of LLDP seemed to be more work than the benefit of "reusing the existing protocol" LLDP for this very simple purpose.

A.3.2. mDNS and L2 support

Multicast DNNS (mDNS) [RFC6762] with DNS Service Discovery (DNS-SD) Resource Records (RRs) as defined in [RFC6763] is a key contender as an ACP discovery protocol. because it relies on link-local IP multicast, it does operates at the subnet level, and is also found in L2 switches. The authors of this document are not aware of mDNS implementation that terminate their mDNS messages on L2 ports instead.
of the subnet level. If mDNS was used as the ACP discovery mechanism on an ACP capable (L3)/L2 switch as outlined in Section 7, then this would be necessary to implement. It is likely that termination of mDNS messages could only be applied to all mDNS messages from such a port, which would then make it necessary to software forward any non-ACP related mDNS messages to maintain prior non-ACP mDNS functionality. Adding support for ACP into such L2 switches with mDNS could therefore create regression problems for prior mDNS functionality on those nodes. With low performance of software forwarding in many L2 switches, this could also make the ACP risky to support on such L2 switches.

A.3.3. Why DULL GRASP

LLDP was not considered because of the above mentioned issues. mDNS was not selected because of the above L2 mDNS considerations and because of the following additional points:

If mDNS was not already existing in a node, it would be more work to implement than DULL GRASP, and if an existing implementation of mDNS was used, it would likely be more code space than a separate implementation of DULL GRASP or a shared implementation of DULL GRASP and GRASP in the ACP.

A.4. Choice of routing protocol (RPL)

This section motivates why RPL - "IPv6 Routing Protocol for Low-Power and Lossy Networks ([RFC6550] was chosen as the default (and in this specification only) routing protocol for the ACP. The choice and above explained profile was derived from a pre-standard implementation of ACP that was successfully deployed in operational networks.

Requirements for routing in the ACP are:

- Self-management: The ACP must build automatically, without human intervention. Therefore routing protocol must also work completely automatically. RPL is a simple, self-managing protocol, which does not require zones or areas; it is also self-configuring, since configuration is carried as part of the protocol (see Section 6.7.6 of [RFC6550]).

- Scale: The ACP builds over an entire domain, which could be a large enterprise or service provider network. The routing protocol must therefore support domains of 100,000 nodes or more, ideally without the need for zoning or separation into areas. RPL has this scale property. This is based on extensive use of default routing.
- Low resource consumption: The ACP supports traditional network infrastructure, thus runs in addition to traditional protocols. The ACP, and specifically the routing protocol must have low resource consumption both in terms of memory and CPU requirements. Specifically, at edge nodes, where memory and CPU are scarce, consumption should be minimal. RPL builds a destination-oriented directed acyclic graph (DODAG), where the main resource consumption is at the root of the DODAG. The closer to the edge of the network, the less state needs to be maintained. This adapts nicely to the typical network design. Also, all changes below a common parent node are kept below that parent node.

- Support for unstructured address space: In the Autonomic Networking Infrastructure, node addresses are identifiers, and may not be assigned in a topological way. Also, nodes may move topologically, without changing their address. Therefore, the routing protocol must support completely unstructured address space. RPL is specifically made for mobile ad-hoc networks, with no assumptions on topologically aligned addressing.

- Modularity: To keep the initial implementation small, yet allow later for more complex methods, it is highly desirable that the routing protocol has a simple base functionality, but can import new functional modules if needed. RPL has this property with the concept of "objective function", which is a plugin to modify routing behavior.

- Extensibility: Since the Autonomic Networking Infrastructure is a new concept, it is likely that changes in the way of operation will happen over time. RPL allows for new objective functions to be introduced later, which allow changes to the way the routing protocol creates the DAGs.

- Multi-topology support: It may become necessary in the future to support more than one DODAG for different purposes, using different objective functions. RPL allow for the creation of several parallel DODAGs, should this be required. This could be used to create different topologies to reach different roots.

- No need for path optimization: RPL does not necessarily compute the optimal path between any two nodes. However, the ACP does not require this today, since it carries mainly non-delay-sensitive feedback loops. It is possible that different optimization schemes become necessary in the future, but RPL can be expanded (see point "Extensibility" above).
A.5. ACP Information Distribution and multicast

IP multicast is not used by the ACP because the ANI (Autonomic Networking Infrastructure) itself does not require IP multicast but only service announcement/discovery. Using IP multicast for that would have made it necessary to develop a zero-touch auto configuring solution for ASM (Any Source Multicast - the original form of IP multicast defined in [RFC1112]), which would be quite complex and difficult to justify. One aspect of complexity where no attempt at a solution has been described in IETF documents is the automatic-selection of routers that should be PIM Sparse Mode (PIM-SM) Rendezvous Points (RPs) (see [RFC7761]). The other aspects of complexity are the implementation of MLD ([RFC4604]), PIM-SM and Anycast-RP (see [RFC4610]). If those implementations already exist in a product, then they would be very likely tied to accelerated forwarding which consumes hardware resources, and that in return is difficult to justify as a cost of performing only service discovery.

Some future ASA may need high performance in-network data replication. That is the case when the use of IP multicast is justified. Such an ASA can then use service discovery from ACP GRASP, and then they do not need ASM but only SSM (Source Specific Multicast, see [RFC4607]) for the IP multicast replication. SSM itself can simply be enabled in the Data-Plane (or even in an update to the ACP) without any other configuration than just enabling it on all nodes and only requires a simpler version of MLD (see [RFC5790]).

LSP (Link State Protocol) based IGP routing protocols typically have a mechanism to flood information, and such a mechanism could be used to flood GRASP objectives by defining them to be information of that IGP. This would be a possible optimization in future variations of the ACP that do use an LSP routing protocol. Note though that such a mechanism would not work easily for GRASP M_DISCOVERY messages which are intelligently (constrained) flooded not across the whole ACP, but only up to a node where a responder is found. We do expect that many future services in ASA will have only few consuming ASA, and for those cases, M_DISCOVERY is the more efficient method than flooding across the whole domain.

Because the ACP uses RPL, one desirable future extension is to use RPLs existing notion of loop-free distribution trees (DODAG) to make GRASPs flooding more efficient both for M_FLOOD and M_DISCOVERY) See Section 6.12.5 how this will be specifically beneficial when using NBMA interfaces. This is not currently specified in this document because it is not quite clear yet what exactly the implications are to make GRASP flooding depend on RPL DODAG convergence and how difficult it would be to let GRASP flooding access the DODAG information.
A.6. Extending ACP channel negotiation (via GRASP)

The mechanism described in the normative part of this document to support multiple different ACP secure channel protocols without a single network wide MTI protocol is important to allow extending secure ACP channel protocols beyond what is specified in this document, but it will run into problem if it would be used for multiple protocols:

The need to potentially have multiple of these security associations even temporarily run in parallel to determine which of them works best does not support the most lightweight implementation options.

The simple policy of letting one side (Alice) decide what is best may not lead to the mutual best result.

The two limitations can easier be solved if the solution was more modular and as few as possible initial secure channel negotiation protocols would be used, and these protocols would then take on the responsibility to support more flexible objectives to negotiate the mutually preferred ACP security channel protocol.

IKEv2 is the IETF standard protocol to negotiate network security associations. It is meant to be extensible, but it is unclear whether it would be feasible to extend IKEv2 to support possible future requirements for ACP secure channel negotiation:

Consider the simple case where the use of native IPsec vs. IPsec via GRE is to be negotiated and the objective is the maximum throughput. Both sides would indicate some agreed upon performance metric and the preferred encapsulation is the one with the higher performance of the slower side. IKEv2 does not support negotiation with this objective.

Consider DTLS and some form of MacSec are to be added as negotiation options - and the performance objective should work across all IPsec, DTLS and MacSec options. In the case of MacSEC, the negotiation would also need to determine a key for the peering. It is unclear if it would be even appropriate to consider extending the scope of negotiation in IKEv2 to those cases. Even if feasible to define, it is unclear if implementations of IKEv2 would be eager to adopt those type of extension given the long cycles of security testing that necessarily goes along with core security protocols such as IKEv2 implementations.

A more modular alternative to extending IKEv2 could be to layer a modular negotiation mechanism on top of the multitude of existing or possible future secure channel protocols. For this, GRASP over TLS could be considered as a first ACP secure channel negotiation...
protocol. The following are initial considerations for such an approach. A full specification is subject to a separate document:

To explicitly allow negotiation of the ACP channel protocol, GRASP over a TLS connection using the GRASP_LISTEN_PORT and the nodes and peers link-local IPv6 address is used. When Alice and Bob support GRASP negotiation, they do prefer it over any other non-explicitly negotiated security association protocol and should wait trying any non-negotiated ACP channel protocol until after it is clear that GRASP/TLS will not work to the peer.

When Alice and Bob successfully establish the GRASP/TLS session, they will negotiate the channel mechanism to use using objectives such as performance and perceived quality of the security. After agreeing on a channel mechanism, Alice and Bob start the selected Channel protocol. Once the secure channel protocol is successfully running, the GRASP/TLS connection can be kept alive or timed out as long as the selected channel protocol has a secure association between Alice and Bob. When it terminates, it needs to be re-negotiated via GRASP/TLS.

Notes:

- Negotiation of a channel type may require IANA assignments of code points.

- TLS is subject to reset attacks, which IKEv2 is not. Normally, ACP connections (as specified in this document) will be over link-local addresses so the attack surface for this one issue in TCP should be reduced (note that this may not be true when ACP is tunneled as described in Section 8.2.2.

- GRASP packets received inside a TLS connection established for GRASP/TLS ACP negotiation are assigned to a separate GRASP domain unique to that TLS connection.

A.7. CAs, domains and routing subdomains

There is a wide range of setting up different ACP solution by appropriately using CAs and the domain and rsusb elements in the domain information field of the domain certificate. We summarize these options here as they have been explained in different parts of the document in before and discuss possible and desirable extensions:

An ACP domain is the set of all ACP nodes using certificates from the same CA using the same domain field. GRASP inside the ACP is run across all transitively connected ACP nodes in a domain.
The rsub element in the domain information field permits the use of addresses from different ULA prefixes. One use case is to create multiple physical networks that initially may be separated with one ACP domain but different routing subdomains, so that all nodes can mutual trust their ACP domain certificates (not depending on rsub) and so that they could connect later together into a contiguous ACP network.

One instance of such a use case is an ACP for regions interconnected via a non-ACP enabled core, for example due to the absence of product support for ACP on the core nodes. ACP connect configurations as defined in this document can be used to extend and interconnect those ACP islands to the NOC and merge them into a single ACP when later that product support gap is closed.

Note that RPL scales very well. It is not necessary to use multiple routing subdomains to scale ACP domains in a way it would be possible if other routing protocols where used. They exist only as options for the above mentioned reasons.

If different ACP domains are to be created that should not allow to connect to each other by default, these ACP domains simply need to have different domain elements in the domain information field. These domain elements can be arbitrary, including subdomains of one another: Domains "example.com" and "research.example.com" are separate domains if both are domain elements in the domain information element of certificates.

It is not necessary to have a separate CA for different ACP domains; an operator can use a single CA to sign certificates for multiple ACP domains that are not allowed to connect to each other because the checks for ACP adjacencies includes comparison of the domain part.

If multiple independent networks choose the same domain name but had their own CA, these would not form a single ACP domain because of CA mismatch. Therefore there is no problem in choosing domain names that are potentially also used by others. Nevertheless it is highly recommended to use domain names that one can have high probability to be unique. It is recommended to use domain names that start with a DNS domain names owned by the assigning organization and unique within it. For example "acp.example.com" if you own "example.com".

A.8. Intent for the ACP

Intent is the architecture component of autonomic networks according to [I-D.ietf-anima-reference-model] that allows operators to issue policies to the network. In a simple instance, Intent could simply
be policies flooded across ACP GRASP and interpreted on every ACP node.

One concern for future definitions of Intent solutions is the problem of circular dependencies when expressing Intent policies about the ACP itself.

For example, Intent could indicate the desire to build an ACP across all domains that have a common parent domain (without relying on the rsub/routing-subdomain solution defined in this document). For example ACP nodes with domain "example.com", "access.example.com", "core.example.com" and "city.core.example.com" should all establish one single ACP.

If each domain has its own source of Intent, then the Intent would simply have to allow adding the peer domains trust anchors (CA) and domain names to the ACP domain membership check (Section 6.1.2) so that nodes from those other domains are accepted as ACP peers.

If this Intent was to be originated only from one domain, it could likely not be made to work because the other domains will not build any ACP connection amongst each other, whether they use the same or different CA due to the ACP domain membership check.

If the domains use the same CA one could change the ACP setup to permit for the ACP to be established between two ACP nodes with different acp-domain-names, but only for the purpose of disseminating limited information, such as Intent, but not to set up full ACP connectivity, specifically not RPL routing and passing of arbitrary GRASP information. Unless the Intent policies permit this to happen across domain boundaries.

This type of approach where the ACP first allows Intent to operate and only then sets up the rest of ACP connectivity based on Intent policy could also be used to enable Intent policies that would limit functionality across the ACP inside a domain, as long as no policy would disturb the distribution of Intent. For example to limit reachability across the ACP to certain type of nodes or locations of nodes.

A.9. Adopting ACP concepts for other environments

The ACP as specified in this document is very explicit about the choice of options to allow interoperable implementations. The choices made may not be the best for all environments, but the concepts used by the ACP can be used to build derived solutions:
The ACP specifies the use of ULA and deriving its prefix from the domain name so that no address allocation is required to deploy the ACP. The ACP will equally work not using ULA but any other /48 IPv6 prefix. This prefix could simply be a configuration of the ACP registrars (for example when using BRSKI) to enroll the domain certificates - instead of the ACP registrar deriving the /48 ULA prefix from the AN domain name.

Some solutions may already have an auto-addressing scheme, for example derived from existing unique device identifiers (e.g., MAC addresses). In those cases it may not be desirable to assign addresses to devices via the ACP address information field in the way described in this document. The certificate may simply serve to identify the ACP domain, and the address field could be empty/unused. The only fix required in the remaining way the ACP operate is to define another element in the domain certificate for the two peers to decide who is Alice and who is Bob during secure channel building. Note though that future work may leverage the acp address to authenticate "ownership" of the address by the device. If the address used by a device is derived from some pre-existing permanent local ID (such as MAC address), then it would be useful to store that address in the certificate using the format of the access address information field or in a similar way.

The ACP is defined as a separate VRF because it intends to support well managed networks with a wide variety of configurations. Therefore, reliable, configuration-indestructible connectivity cannot be achieved from the Data-Plane itself. In solutions where all transit connectivity impacting functions are fully automated (including security), indestructible and resilient, it would be possible to eliminate the need for the ACP to be a separate VRF. Consider the most simple example system in which there is no separate Data-Plane, but the ACP is the Data-Plane. Add BRSKI, and it becomes a fully autonomic network - except that it does not support automatic addressing for user equipment. This gap can then be closed for example by adding a solution derived from [I-D.ietf-anima-prefix-management].

TCP/TLS as the protocols to provide reliability and security to GRASP in the ACP may not be the preferred choice in constrained networks. For example, CoAP/DTLS (Constrained Application Protocol) may be preferred where they are already used, allowing to reduce the additional code space footprint for the ACP on those devices. Hop-by-hop reliability for ACP GRASP messages could be made to support protocols like DTLS by adding the same type of negotiation as defined in this document for ACP secure channel protocol negotiation. End-to-end GRASP connections can be made to select their transport protocol in future extensions of the ACP meant to better support
constrained devices by indicating the supported transport protocols (e.g.: TLS/DTLS) via GRASP parameters of the GRASP objective through which the transport endpoint is discovered.

The routing protocol chosen by the ACP design (RPL) does explicitly not optimize for shortest paths and fastest convergence. Variations of the ACP may want to use a different routing protocol or introduce more advanced RPL profiles.

Variations such as what routing protocol to use, or whether to instantiate an ACP in a VRF or (as suggested above) as the actual Data-Plane, can be automatically chosen in implementations built to support multiple options by deriving them from future parameters in the certificate. Parameters in certificates should be limited to those that would not need to be changed more often than certificates would need to be updated anyhow; Or by ensuring that these parameters can be provisioned before the variation of an ACP is activated in a node. Using BRSKI, this could be done for example as additional follow-up signaling directly after the certificate enrollment, still leveraging the BRSKI TLS connection and therefore not introducing any additional connectivity requirements.

Last but not least, secure channel protocols including their encapsulations are easily added to ACP solutions. ACP hop-by-hop network layer secure channels could also be replaced by end-to-end security plus other means for infrastructure protection. Any future network OAM should always use end-to-end security anyhow and can leverage the domain certificates and is therefore not dependent on security to be provided for by ACP secure channels.

A.10. Further options / futures

A.10.1. Auto-aggregation of routes

Routing in the ACP according to this specification only leverages the standard RPL mechanism of route optimization, e.g. keeping only routes that are not towards the RPL root. This is known to scale to networks with 20,000 or more nodes. There is no auto-aggregation of routes for /48 ULA prefixes (when using rsub in the domain information field) and/or Zone-ID based prefixes.

Automatic assignment of Zone-ID and auto-aggregation of routes could be achieved for example by configuring zone-boundaries, announcing via GRASP into the zones the zone parameters (zone-ID and /48 ULA prefix) and auto-aggregating routes on the zone-boundaries. Nodes would assign their Zone-ID and potentially even /48 prefix based on the GRASP announcements.
A.10.2. More options for avoiding IPv6 Data-Plane dependency

As described in Section 6.12.2, the ACP depends on the Data-Plane to establish IPv6 link-local addressing on interfaces. Using a separate MAC address for the ACP allows to fully isolate the ACP from the data-plane in a way that is compatible with this specification. It is also an ideal option when using Single-root input/output virtualization (SR-IOV - see https://en.wikipedia.org/wiki/Single-root_input/output_virtualization [2]) in an implementation to isolate the ACP because different SR-IOV interfaces use different MAC addresses.

When additional MAC address(es) are not available, separation of the ACP could be done at different demux points. The same subnet interface could have a separate IPv6 interface for the ACP and Data-Plane and therefore separate link-local addresses for both, where the ACP interface is non-configurable on the Data-Plane. This too would be compatible with this specification and not impact interoperability.

An option that would require additional specification is to use a different Ethertype from 0x86DD (IPv6) to encapsulate IPv6 packets for the ACP. This would be a similar approach as used for IP authentication packets in [IEEE-802.1X] which use the Extensible Authentication Protocol over Local Area Network (EAPoL) ethertype (0x88A2).

Note that in the case of ANI nodes, all the above considerations equally apply to the encapsulation of BRSKI packets including GRASP used for BRSKI.

A.10.3. ACP APIs and operational models (YANG)

Future work should define YANG ([RFC7950]) data model and/or node internal APIs to monitor and manage the ACP.

Support for the ACP Adjacency Table (Section 6.2) and ACP GRASP need to be included into such model/API.

A.10.4. RPL enhancements
The profile for RPL specified in this document builds only one spanning-tree path set to a root (NOC). In the presence of multiple NOCs, routing toward the non-root NOCs may be suboptimal. Figure 17 shows an extreme example. Assuming that node ACP1 becomes the RPL root, traffic between ACP11 and NOC2 will pass through ACP4-ACP3-ACP1-ACP2 instead of ACP4-ACP2 because the RPL calculated DODAG/routes are shortest paths towards the RPL root.

To overcome these limitations, extensions/modifications to the RPL profile can provide optimality for multiple NOCs. This requires utilizing Data-Plane artifact including IPinIP encap/decap on ACP routers and processing of IPv6 RPI headers. Alternatively, (Src,Dst) routing table entries could be used.

Flooding of ACP GRASP messages can be further constrained and therefore optimized by flooding only via links that are part of the RPL DODAG.

A.10.5. Role assignments

ACP connect is an explicit mechanism to "leak" ACP traffic explicitly (for example in a NOC). It is therefore also a possible security gap when it is easy to enable ACP connect on arbitrary compromised ACP nodes.

One simple solution is to define an extension in the ACP certificates ACP information field indicating the permission for ACP connect to be configured on that ACP node. This could similarly be done to decide whether a node is permitted to be a registrar or not.
Tying the permitted "roles" of an ACP node to the ACP domain certificate provides fairly strong protection against misconfiguration, but is still subject to code modifications.

Another interesting role to assign to certificates is that of a NOC node. This would allow to limit certain type of connections such as OAM TLS connections to only NOC initiator or responders.

A.10.6.  Autonomic L3 transit

In this specification, the ACP can only establish autonomic connectivity across L2 hops and only explicitly configured options to tunnel across L3. Future work should specify mechanisms to automatically tunnel ACP across L3 networks. A hub&spoke option would allow to tunnel across the Internet to a cloud or central instance of the ACP, a peer-to-peer tunneling mechanism could tunnel ACP islands across an L3VPN infrastructure.

A.10.7.  Diagnostics

Section 10.1 describes diagnostics options that can be done without changing the external, interoperability affecting characteristics of ACP implementations.

Even better diagnostics of ACP operations is possible with additional signaling extensions, such as:

1. Consider if LLDP should be a recommended functionality for ANI devices to improve diagnostics, and if so, which information elements it should signal (insecure). Includes potentially new information elements.

2. In alternative to LLDP, A DULL GRASP diagnostics objective could be defined to carry these information elements.

3. The IDevID of BRSKI pledges should be included in the selected insecure diagnostics option.

4. A richer set of diagnostics information should be made available via the secured ACP channels, using either single-hop GRASP or network wide "topology discovery" mechanisms.

A.10.8.  Avoiding and dealing with compromised ACP nodes

Compromised ACP nodes pose the biggest risk to the operations of the network. The most common type of compromise is leakage of credentials to manage/configure the device and the application of malicious configuration including the change of access credentials,
but not the change of software. Most of today's networking equipment
should have secure boot/software infrastructure anyhow, so attacks
that introduce malicious software should be a lot harder.

The most important aspect of security design against these type of
attacks is to eliminate password based configuration access methods
and instead rely on certificate based credentials handed out only to
nodes where it is clear that the private keys can not leak. This
limits unexpected propagation of credentials.

If password based credentials to configure devices still need to be
supported, they must not be locally configurable, but only be
remotely provisioned or verified (through protocols like Radius or
Diameter), and there must be no local configuration permitting to
change these authentication mechanisms, but ideally they should be
autoconfiguring across the ACP. See
[I-D.eckert-anima-noc-autoconfig].

Without physical access to the compromised device, attackers with
access to configuration should not be able to break the ACP
connectivity, even when they can break or otherwise manipulate
(spoof) the data-plane connectivity through configuration. To
achieve this, it is necessary to avoid providing configuration
options for the ACP, such as enabling/disabling it on interfaces.
For example there could be an ACP configuration that locks down the
current ACP config unless factory reset is done.

With such means, the valid administration has the best chances to
maintain access to ACP nodes, discover malicious configuration though
ongoing configuration tracking from central locations for example,
and to react accordingly.

The primary reaction is withdrawal/change of credentials, terminate
malicious existing management sessions and fixing the configuration.
Ensuring that management sessions using invalidated credentials are
terminated automatically without recourse will likely require new
work.

Only when these steps are not feasible would it be necessary to
revoke or expire the ACP domain certificate credentials and consider
the node kicked off the network - until the situation can be further
rectified, likely requiring direct physical access to the node.

Without extensions, compromised ACP nodes can only be removed from
the ACP at the speed of CRL/OCSP information refresh or expiry (and
non-removal) of short lived certificates. Future extensions to the
ACP could for example use GRASP flooding distribution of triggered
updates of CRL/OCSP or explicit removal indication of the compromised nodes domain certificate.

Authors’ Addresses

Toerless Eckert (editor)
Huawei USA - Futurewei Technologies Inc.
2330 Central Expy
Santa Clara   95050
USA

Email: tte+ietf@cs.fau.de

Michael H. Behringer (editor)
Email: michael.h.behringer@gmail.com

Steinthor Bjarnason
Arbor Networks
2727 South State Street, Suite 200
Ann Arbor   MI 48104
United States

Email: sbjarnason@arbor.net
Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-22

Abstract

This document specifies automated bootstrapping of an Autonomic Control Plane. To do this a remote secure key infrastructure (BRSKI) is created using manufacturer installed X.509 certificate, in combination with a manufacturer’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.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 19, 2019.
# Table of Contents

1. Introduction ................................................. 4  
   1.1. Prior Bootstrapping Approaches ....................... 6  
   1.2. Terminology ........................................... 7  
   1.3. Scope of solution ..................................... 10  
      1.3.1. Support environment .............................. 10  
      1.3.2. Constrained environments ......................... 10  
      1.3.3. Network Access Controls .......................... 11  
      1.3.4. Bootstrapping is not Booting ..................... 11  
   1.4. Leveraging the new key infrastructure / next steps ... 11  
   1.5. Requirements for Autonomic Network Infrastructure (ANI) devices ............................................. 12  
2. Architectural Overview ..................................... 12  
   2.1. Behavior of a Pledge ................................. 14  
   2.2. Secure Imprinting using Vouchers .................... 15  
   2.3. Initial Device Identifier ............................. 16  
      2.3.1. Identification of the Pledge ....................... 16  
      2.3.2. MASA URI extension ................................. 17  
   2.4. Protocol Flow ......................................... 19  
   2.5. Architectural Components ............................. 21  
      2.5.1. Pledge ............................................ 21  
      2.5.2. Join Proxy ....................................... 21  
      2.5.3. Domain Registrar ................................ 21  
      2.5.4. Manufacturer Service .............................. 21  
      2.5.5. Public Key Infrastructure (PKI) .................. 21  
   2.6. Certificate Time Validation ......................... 22  
      2.6.1. Lack of realtime clock ........................... 22  
      2.6.2. Infinite Lifetime of IDevID ....................... 22  
   2.7. Cloud Registrar ...................................... 22  
   2.8. Determining the MASA to contact ..................... 23  
3. Voucher-Request artifact ................................. 23  
   3.1. Nonceless Voucher Requests ......................... 24
3.2. Tree Diagram ........................................... 24
3.3. Examples ................................................. 25
3.4. YANG Module ............................................. 26
4. Proxying details (Pledge - Proxy - Registrar) ................. 29
  4.1. Pledge discovery of Proxy ............................... 30
  4.1.1. Proxy GRASP announcements .......................... 32
  4.2. CoAP connection to Registrar ........................... 33
  4.3. Proxy discovery and communication of Registrar ........ 33
5. Protocol Details (Pledge - Registrar - MASA) ................. 34
  5.1. BRSKI-EST TLS establishment details ................ 36
  5.2. Pledge Requests Voucher from the Registrar ........... 36
  5.3. Registrar Authorization of
       Pledge .................................................. 37
  5.4. BRSKI-MASA TLS establishment details ................ 38
  5.5. Registrar Requests Voucher from MASA ................. 39
      5.5.1. MASA renewal of expired vouchers ................ 40
      5.5.2. MASA verification of voucher-request signature consistency ................. 41
      5.5.3. MASA authentication of registrar (certificate) ... 41
      5.5.4. MASA revocation checking of registrar (certificate) . 41
      5.5.5. MASA verification of pledge prior-signed-voucher-request ................. 41
      5.5.6. MASA pinning of registrar ......................... 42
      5.5.7. MASA nonce handling .............................. 42
  5.6. MASA and Registrar Voucher Response .................... 42
      5.6.1. Pledge voucher verification ....................... 45
      5.6.2. Pledge authentication of provisional TLS connection . 45
  5.7. Pledge BRSKI Status Telemetry ........................ 46
  5.8. Registrar audit log request ............................ 47
      5.8.1. MASA audit log response .......................... 48
      5.8.2. Registrar audit log verification .................. 49
  5.9. EST Integration for PKI bootstrapping ................ 50
      5.9.1. EST Distribution of CA Certificates ................ 51
      5.9.2. EST CSR Attributes ................................ 51
      5.9.3. EST Client Certificate Request ................... 52
      5.9.4. Enrollment Status Telemetry ....................... 52
      5.9.5. Multiple certificates ............................. 53
      5.9.6. EST over CoAP .................................... 53
  6. Clarification of transfer-encoding ......................... 54
  7. Reduced security operational modes ......................... 54
      7.1. Trust Model ......................................... 54
      7.2. Pledge security reductions .......................... 55
      7.3. Registrar security reductions ........................ 56
      7.4. MASA security reductions ................................ 57
  8. IANA Considerations ...................................... 57
      8.1. Well-known EST registration ........................ 58
      8.2. PKIX Registry ....................................... 58
      8.3. Pledge BRSKI Status Telemetry ....................... 58
1. Introduction

BRSKI provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices that are called pledges in this document.

This document primarily provides for the needs of the ISP and Enterprise focused ANIMA Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. Other users of the BRSKI protocol will need to provide separate applicability statements that include privacy and security considerations appropriate to that
This document describes how pledges discover (or be discovered by) an element of the network domain to which the pledge belongs to perform the bootstrap. This element (device) is called the registrar.

Before any other operation, pledge and registrar need to establish mutual trust:

1. Registrar authenticating the pledge: "Who is this device? What is its identity?"
2. Registrar authorizing the pledge: "Is it mine? Do I want it? What are the chances it has been compromised?"
3. Pledge authenticating the registrar: "What is this registrar’s identity?"
4. Pledge authorizing the registrar: "Should I join it?"

This document details protocols and messages to answer the above questions. It uses a TLS connection and an PKIX (X.509v3) certificate (an IEEE 802.1AR [IDevID] LDevID) of the pledge to answer points 1 and 2. It uses a new artifact called a "voucher" that the registrar receives from a "Manufacturer Authorized Signing Authority" and passes to the pledge to answer points 3 and 4.

A proxy provides very limited connectivity between the pledge and the registrar.

The syntactic details of vouchers are described in detail in [RFC8366]. This document details automated protocol mechanisms to obtain vouchers, including the definition of a ‘voucher-request’ message that is a minor extension to the voucher format (see Section 3) defined by [RFC8366].

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 that 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. Prior 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 (often pre-shared keys, including mechanisms like SIM cards) is pre-provisioned on each new device in a costly and non-scalable manner. Existing automated mechanisms are known as non-secured 'Trust on First Use' (TOFU) [RFC7435], 'resurrecting duckling' [Stajano99theresurrecting] or 'pre-staging'.

Another prior approach has been to try and minimize user actions during bootstrapping, but not eliminate all user-actions. The original EST protocol [RFC7030] does reduce user actions during bootstrap but does not provide solutions for how the following protocol steps can be made autonomic (not involving user actions):

- using the Implicit Trust Anchor [RFC7030] database to authenticate an owner specific service (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 manufacturer service using a common client-server authentication model. After mutual authentication, appropriate credentials to authenticate the target domain are transferred to the pledge. This creates serveral problems and limitations:
o the pledge requires realtime connectivity to the manufacturer service,

o the domain identity is exposed to the manufacturer service (this is a privacy concern),

o the manufacturer 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 pinned-domain-cert leaf, i.e. the Registrars’ certificate. This is consistent with the subject key identifier (Section 4.2.1.2 [RFC5280]).

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 at the factory.

Voucher: A signed artifact from the MASA 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 is asserted. Multiple voucher types are defined in [RFC8366]

Domain: The set of entities that share a common local 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 manages the private key 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". Within [I-D.ietf-anima-reference-model] this is referred to as the "join registrar autonomic service agent". Other communities use the abbreviation "JRC".

(Public) Key Infrastructure: The collection of systems and processes that sustain the activities of a public key system. The registrar acts as an [RFC5280] and [RFC5272] (see section 7) "Registration Authority".

Join Proxy: A domain entity that helps the pledge join the domain. A join 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. For simplicity this document sometimes uses the term of 'proxy' to indicate the join proxy. The pledge is unaware that they are communicating with a proxy rather than directly with a registrar.

Circuit Proxy: A stateful implementation of the join proxy. This is the assumed type of proxy.

IPIP Proxy: A stateless proxy alternative.
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 [RFC8366]

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.

nonced: a voucher (or request) that contains a nonce (the normal case).

nonceless: a voucher (or request) that does not contain a nonce, relying upon accurate clocks for expiration, or which does not expire.

manufacturer: the term manufacturer is used throughout this document to be the entity that created the device. This is typically the "original equipment manufacturer" or OEM, but in more complex situations it could be a "value added retailer" (VAR), or possibly even a systems integrator. In general, it a goal of BRSKI to eliminate small distinctions between different sales channels. The reason for this is that it permits a single device, with a uniform firmware load, to be shipped directly to all customers. This eliminates costs for the manufacturer. This also reduces the number of products supported in the field increasing the chance that firmware will be more up to date.

ANI: The Autonomic Network Infrastructure as defined by [I-D.ietf-anima-reference-model]. This document details specific requirements for pledges, proxies and registrars when they are part of an ANI.
offline: When an architectural component cannot perform realtime communications with a peer, either due to network connectivity or because the peer is turned off, the operation is said to be occurring offline.

1.3. Scope of solution

1.3.1. Support environment

This solution (BRSKI) can support large router platforms with multi-gigabit inter-connections, mounted in controlled access data centers. But this solution is not exclusive to large equipment: 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 a 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 its presence through broadcasting.

Nomadic or mobile devices often need to acquire credentials to access the network at the new location. An example of this is mobile phone roaming among network operators, or even between cell towers. This is usually called handoff. BRSKI does not provide a low-latency handoff which is usually a requirement in such situations. For these solutions BRSKI can be used to create a relationship (an LDevID) with the "home" domain owner. The resulting credentials are then used to provide credentials more appropriate for a low-latency handoff.

1.3.2. Constrained environments

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).
Specifically, there are protocol aspects described here that 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 that could be inappropriate.

1.3.3. Network Access Controls

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 consistent 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.

1.3.4. Bootstrapping is not Booting

This document describes "bootstrapping" as the protocol used to obtain a local trust anchor. It is expected that this trust anchor, along with any additional configuration information subsequently installed, is persisted on the device across system restarts ("booting"). Bootstrapping occurs only infrequently such as when a device is transferred to a new owner or has been reset to factory default settings.

1.4. Leveraging the new key infrastructure / next steps

As a result of the protocol described herein, the bootstrapped devices have the Domain CA trust anchor in common. An end entity certificate has optionally been issued from the Domain CA. This
makes it possible to securely deploy functionalities across the domain, e.g:

- 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]).

1.5. Requirements for Autonomic Network Infrastructure (ANI) devices

The BRSKI protocol can be used in a number of environments. Some of the options in this document is the result of requirements that are out of the ANI scope. This section defines the base requirements for ANI devices.

For devices that intend to become part of an Autonomic Network Infrastructure (ANI) ([I-D.ietf-anima-reference-model]) that includes an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]), the BRSKI protocol MUST be implemented.

The pledge must perform discovery of the proxy as described in Section 4.1 using GRASP M_FLOOD announcements.

Upon successfully validating a voucher artiface, a status telemetry MUST be returned. See Section 5.7.

An ANIMA ANI pledge MUST implement the EST automation extensions described in Section 5.9. They supplement the [RFC7030] EST to better support automated devices that do not have an end user.

The ANI Join Registrar ASA MUST support all the BRSKI and above listed EST operations.

All ANI devices SHOULD support the BRSKI proxy function, using circuit proxies over the ACP. (See Section 4.3)

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components.
We assume a multi-vendor network. In such an environment there could be a Manufacturer Service for each manufacturer that supports devices following this document’s specification, or an integrator could provide a generic service authorized by multiple manufacturers. It is unlikely that an integrator could provide Ownership Tracking services for multiple manufacturers 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 domain provides initial device connectivity sufficient for bootstrapping through a proxy. The domain registrar authenticates the pledge, makes authorization decisions, and distributes vouchers obtained from the Manufacturer 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.

```
|
+------v------+
|     (1) Discover     |
+---------------------+
| (2) Identity        |
| rejected             |
| (3) Request Join    |
| (4) Imprint Bad MASA response | send Voucher Status Telemetry |
| (5) Enroll Failure | \___/ (e.g. 201 ’Retry-After’) |
|                       |
| Enroll Failure       |
|
|
|
```

Figure 2: pledge state diagram

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. Request to join the discovered registrar. A unique nonce is included ensuring that any responses can be associated with this particular bootstrapping attempt.

4. Imprint on the registrar. This requires verification of the manufacturer service provided voucher. A voucher contains sufficient information for the pledge to complete authentication of a registrar. This document details this step in depth.

5. Enroll. After imprint an authenticated TLS (HTTPS) connection exists between pledge and registrar. Enrollment over Secure Transport (EST) [RFC7030] is then used to obtain a domain certificate from a registrar.

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.

This specification details integration with EST enrollment so that pledges can optionally obtain a locally issued certificate, although any REST interface could be integrated in future work.

2.2. Secure Imprinting using Vouchers

A voucher is a cryptographically protected artifact (a digital signature) 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 [RFC8366].

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 can indiscriminately issue vouchers and log claims of ownership by domains. At the highest security levels issuance of vouchers can be integrated with complex sales channel integrations that are beyond the scope of this document. The sales channel integration would verify actual (legal) ownership of the pledge by the domain. 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 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 among 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 pledge voucher-request signing is via a PKIX certificate installed during the manufacturing process. This is the 802.1AR Initial Device Identifier (IDevID), and it provides a basis for authenticating the pledge during the protocol exchanges described here. There is no requirement for a common root PKI hierarchy. Each device manufacturer can generate its own root certificate. Specifically, the IDevID enables:

1. Uniquely identifying the pledge by the Distinguished Name (DN) and subjectAltName (SAN) parameters in the IDevID. The unique identification of a pledge in the voucher objects are derived from those parameters as described below.

2. Provides a cryptographic authentication of the pledge to the Registrar (see Section 5.3).

3. Secure auto-discovery of the pledge’s MASA by the registrar (see Section 2.8).

4. Signing of voucher-request by the pledge’s IDevID (see Section 3).

5. Provides a cryptographic authentication of the pledge to the MASA (see Section 5.5.5).

Section 7.2.13 of [IDevID] discusses keyUsage and extendedKeyUsage extensions in the IDevID certificate. Any restrictions included reduce the utility of the IDevID and so this specification RECOMMENDS that no key usage restrictions be included. Additionally, [RFC5280] section 4.2.1.3 does not require key usage restrictions for end entity certificates.

2.3.1. Identification of the Pledge

In the context of BRSKI, pledges are uniquely identified by a "serial-number". This serial-number is used both in the "serial-number" field of voucher or voucher-requests (see Section 3) and in local policies on registrar or MASA (see Section 5).
The following fields are defined in [IDevID] and [RFC5280]:

- The subject field’s DN encoding MUST include the "serialNumber" attribute with the device’s unique serial number. (from [IDevID] section 7.2.8, and [RFC5280] section 4.1.2.4’s list of standard attributes)

- The subject-alt field’s encoding MAY include a non-critical version of the RFC4108 defined HardwareModuleName. (from [IDevID] section 7.2.9) If the IDevID is stored in a Trusted Platform Module (TPM), then this field MAY contain the TPM identification rather than the device’s serial number. If both fields are present, then the subject field takes precedence.

and they are used as follows by the pledge to build the "serial-number" that is placed in the voucher-request. In order to build it, the fields need to be converted into a serial-number of "type string". The following methods are used depending on the first available IDevID certificate field (attempted in this order):

1. [RFC4519] section 2.31 provides an example ("WI-3005") of the Distinguished Name "serialNumber" attribute. [RFC4514] indicates this is a printable string so no encoding is necessary.

2. The HardwareModuleName hwSerialNum OCTET STRING. This value is base64 encoded to convert it to a printable string format.

The above process to locate the serial-number MUST be performed by the pledge when filling out the voucher-request. Signed voucher-requests are always passed up to the MASA.

As explained in Section 5.5 the Registrar MUST extract the serial-number again itself from the pledge’s TLS certificate. It can consult the serial-number in the pledge-request if there are any possible confusion about the source of the serial-number (hwSerialNum vs serialNumber).

2.3.2. MASA URI extension

This document defines a new PKIX non-critical certificate extension to carry the MASA URI. This extension is intended to be used in the IDevID certificate. 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 IRI provides the authority...
information. The BRSKI "/.well-known" tree ([RFC5785]) is described in Section 5.

As explained in [RFC5280] section 7.4, a complete IRI SHOULD be in this extension, including the scheme, iauthority, and ipath. As a consideration to constrained systems, this MAY be reduced to only the iauthority, in which case a scheme of "https://" and ipath of "/.well-known/est" is to be assumed, as explained in section Section 5.

The registry can assume that only the iauthority is present in the extension, if there are no slash ("/") characters in the extension.

Section 7.4 of [RFC5280] calls out various schemes that MUST be supported, including ldap, http and ftp. However, the registrar MUST use https for the BRSKI-MASA connection.

The new extension is identified as follows:
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:
Figure 3

2.5. Architectural Components

2.5.1. Pledge

The pledge is the device that is attempting to join. Until the pledge completes the enrollment process, it has link-local network connectivity only to the proxy.

2.5.2. Join Proxy

The join proxy provides HTTPS connectivity between the pledge and the registrar. A circuit proxy mechanism is described in Section 4. Additional mechanisms, including a CoAP mechanism and a stateless IPIP mechanism are the subject of future work.

2.5.3. Domain Registrar

The domain's registrar operates as the BRSKI-MASA client when requesting vouchers from the MASA (see Section 5.4). The registrar operates as the BRSKI-EST server when pledges request vouchers (see Section 5.1). The registrar operates as the BRSKI-EST server "Registration Authority" if the pledge requests an end entity certificate over the BRSKI-EST connection (see Section 5.9).

The registrar uses an Implicit Trust Anchor database for authenticating the BRSKI-MASA TLS connection MASA certificate. The registrar uses a different Implicit Trust Anchor database for authenticating the BRSKI-EST TLS connection pledge client certificate. Configuration or distribution of these trust anchor databases is out-of-scope of this specification.

2.5.4. Manufacturer Service

The Manufacturer Service provides two logically separate functions: the Manufacturer Authorized Signing Authority (MASA) described in Section 5.5 and Section 5.6, and an ownership tracking/auditing function described in Section 5.7 and Section 5.8.

2.5.5. Public Key Infrastructure (PKI)

The Public Key Infrastructure (PKI) administers certificates for the domain of concerns, providing the trust anchor(s) for it and allowing enrollment of pledges with domain certificates.

The voucher provides a method for the distribution of a single PKI trust anchor (as the "pinned-domain-cert"). A distribution of the full set of current trust anchors is possible using the optional EST integration.
The domain's registrar acts as an [RFC5272] Registration Authority, requesting certificates for pledges from the Key Infrastructure.

The expectations of the PKI are unchanged from EST [[RFC7030]]. This document does not place any additional architectural requirements on the Public Key Infrastructure.

2.6. Certificate Time Validation

2.6.1. Lack of realtime clock

Many devices when bootstrapping do not have knowledge of the current time. Mechanisms such as Network Time Protocols cannot be secured until bootstrapping is complete. Therefore bootstrapping is defined in a method that does not require knowledge of the current time. A pledge MAY ignore all time stamps in the voucher and in the certificate validity periods if it does not know the current time.

The pledge is exposed to dates in the following five places: registrar certificate notBefore, registrar certificate notAfter, voucher created-on, and voucher expires-on. Additionally, CMS signatures contain a signingTime.

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

2.6.2. InfiniteLifetime of IDevID

[RFC5280] explains that long lived pledge certificates "SHOULD be assigned the GeneralizedTime value of 99991231235959Z". Registrars MUST support such lifetimes and SHOULD support ignoring pledge lifetimes if they did not follow the RFC5280 recommendations.

For example, IDevID may have incorrect lifetime of N <= 3 years, rendering replacement pledges from storage useless after N years unless registrars support ignoring such a lifetime.

2.7. Cloud Registrar

There exist operationally open network wherein devices gain unauthenticated access to the internet at large. In these use cases the management domain for the device needs to be discovered within the larger internet. These are less likely within the anima scope but may be more important in the future.

There are additionally some greenfield situations involving an entirely new installation where a device may have some kind of
management uplink that it can use (such as via 3G network for instance). In such a future situation, the device might use this management interface to learn that it should configure itself to become the local registrar.

In order to support these scenarios, the pledge MAY contact a well known URI of a cloud registrar if a local registrar cannot 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.

2.8. 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 (IDevID) 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 C.

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 manufacturer. Note that the registrar can only select the configured MASA URL based on the trust anchor -- so manufacturers 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

Voucher-requests are how vouchers are requested. The semantics of the vouchers are described below, in the YANG model.

A pledge forms the "pledge voucher-request" and submits it to the registrar.
The registrar in turn forms the "registrar voucher-request", and submits it to the MASA.

The "proximity-registrar-cert" leaf is used in the pledge voucher-requests. This provides a method for the pledge to assert the registrar’s proximity.

The "prior-signed-voucher-request" leaf is used in registrar voucher-requests. If present, it is the signed pledge voucher-request. This provides a method for the registrar to forward the pledge’s signed request to the MASA. This completes transmission of the signed "proximity-registrar-cert" leaf.

Unless otherwise signaled (outside the voucher-request artifact), the signing structure is as defined for vouchers, see [RFC8366].

3.1. Nonceless Voucher Requests

A registrar MAY also retrieve nonceless vouchers by sending nonceless voucher-requests to the MASA in order to obtain vouchers for use when the registrar does not have connectivity to the MASA. No "prior-signed-voucher-request" leaf would be included. The registrar will also need to know the serial number of the pledge. This document does not provide a mechanism for the registrar to learn that in an automated fashion. Typically this will be done via scanning of barcode or QR-code on packaging, or via some sales channel integration.

3.2. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher-request document. The voucher-request builds upon the voucher artifact described in [RFC8366]. The tree diagram is described in [RFC8340]. Each node in the diagram is fully described by the YANG module in Section 3.4. Please review the YANG module for a detailed description of the voucher-request format.
module: ietf-voucher-request

grouping 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

3.3. Examples

This section provides voucher-request examples for illustration purposes. For detailed examples, see Appendix D.2. These examples conform to the encoding rules defined in [RFC7951].

Example (1)  The following example illustrates a pledge voucher-request. The assertion leaf is indicated as ‘proximity’ and the registrar’s TLS server certificate is included in the ‘proximity-registrar-cert’ leaf. See Section 5.2.

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

Example (2)  The following example illustrates a registrar voucher-request. The ‘prior-signed-voucher-request’ leaf is populated with the pledge’s voucher-request (such as the prior example). The pledge’s voucher-request is a binary object. In the JSON encoding used here it must be base64 encoded. The nonce, created-on and assertion is carried forward. The serial-number is extracted from the pledge’s Client Certificate from the TLS connection. See Section 5.5.
Example (3) The following example illustrates a registrar 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 can not communicate with the registrar (such as when it is powered down, or still in packaging), and therefore could not submit a nonce. This scenario is most useful when the registrar is aware that it will not be able to reach the MASA during deployment. See Section 5.5.

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

3.4. YANG Module

Following is a YANG [RFC7950] module formally extending the [RFC8366] voucher into a voucher-request.

```
<CODE BEGINS> file "ietf-voucher-request@2018-02-14.yang"
module ietf-voucher-request {
    yang-version 1.1;

    namespace
    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";
    }

    ...  // YANG code

import ietf-voucher {
  prefix v;
  description "This module defines the format for a voucher, which is produced by a pledge’s manufacturer or delegate (MASA) to securely assign a pledge to an 'owner', so that the pledge may establish a secure connection to the owner's network infrastructure";

  reference "RFC YYY: 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 defines the format for a voucher request. It is a superset of the voucher itself. It provides content to the MASA for consideration during a voucher request.

  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 "2018-02-14" {
  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;
    }
    refine "voucher/domain-cert-revocation-checks" {
      description "The domain-cert-revocation-checks field
                     is not valid in a voucher request, and
                     any occurrence MUST be ignored";
    }
    refine "voucher/assertion" {
      mandatory false;
      description "Any assertion included in voucher
                     requests SHOULD be ignored by the MASA.";
    }
  }
  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
For example, a pledge might sign a voucher request with a proximity-registrar-cert, and the registrar then includes it in the prior-signed-voucher-request field. This is a simple mechanism for a chain of trusted parties to change a voucher request, while maintaining the prior signature information.

The Registrar and MASA 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-request 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 a proximity assertion is requested."
}
The proxy does not terminate the TLS handshake: it passes streams of bytes onward without examination. A proxy MUST NOT assume any specific TLS version.

A Registrar can directly provide the proxy announcements described below, in which case the announced port can point directly to the Registrar itself. In this scenario the pledge is unaware that there is no proxing occurring. This is useful for Registrars servicing pledges on directly connected networks.

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

During the discovery of the Registrar by the Join Proxy, the Join Proxy will also learn which kinds of proxy mechanisms are available. This will allow the Join Proxy to use the lowest impact mechanism which the Join Proxy and Registrar have in common.

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. The communication between the pledge is over IPv6 Link-Local addresses.

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. To limit pervasive monitoring ([RFC7258]), a new temporary address MAY use a short lifetime (that is, set TEMP_PREFERRED_LIFETIME to be short). 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.

While the GRASP M_FLOOD mechanism is passive for the pledge, the optional other methods (mDNS, and IPv4 methods) are active. The pledge SHOULD run those methods in parallel with listening to for the M_FLOOD. The active methods SHOULD exponentially back-off to a maximum of one hour to avoid overloading the network with discovery attempts. Detection of change of physical link status (ethernet carrier for instance) SHOULD reset the exponential back off.

The pledge could discover more than one proxy on a given physical interface. The pledge can have a multitude of physical interfaces as well: a layer-2/3 ethernet switch may have hundreds of physical ports.

Each possible proxy offer SHOULD be attempted up to the point where a voucher is received: while there are many ways in which the attempt may fail, it does not succeed until the voucher has been validated.

The connection attempts via a single proxy SHOULD exponentially back-off to a maximum of one hour to avoid overloading the network infrastructure. The back-off timer for each MUST be independent of other connection attempts.

Connection attempts SHOULD be run in parallel to avoid head of queue problems wherein an attacker running a fake proxy or registrar could perform protocol actions intentionally slowly. The pledge SHOULD continue to listen to for additional GRASP M_FLOOD messages during the connection attempts.

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 (assuming that none succeeded) the device MUST return to listening for GRASP M_FLOOD. It SHOULD periodically retry the manufacturer specific mechanisms. The pledge MAY prioritize selection order as appropriate for the anticipated environment.
4.1.1. Proxy GRASP announcements

A proxy uses the DULL GRASP M_FLOOD mechanism to announce itself. This announcement can be within the same message as the ACP announcement detailed in [I-D.ietf-anima-autonomic-control-plane]. The M_FLOOD is formatted as follows:

[M_FLOOD, 12340815, h'fe800000000000000000000000000001', 180000, 
["AN_Proxy", 4, 1, ""],
[O_IPv6_LOCATOR, 
h'fe800000000000000000000000000001', IPPROTO_TCP, 4443]]

Figure 6b: Proxy Discovery

The formal CDDL [I-D.ietf-cbor-cddl] definition is:

flood-message = [M_FLOOD, session-id, initiator, ttl, 
+[objective, (locator-option / [])]]

objective = ["AN_Proxy", objective-flags, loop-count, 
objective-value]

ttl             = 180000     ; 180,000 ms (3 minutes)
initiator = ACP address to contact Registrar
objective-flags   = sync-only  ; as in GRASP spec
sync-only         =  4         ; M_FLOOD only requires synchronization
loop-count        =  1         ; one hop only
objective-value   =  any       ; none
locator-option    = [ O_IPv6_LOCATOR, ipv6-address, 
transport-proto, port-number ]
ipv6-address      = the v6 LL of the Proxy
$transport-proto /= IPPROTO_TCP   ; note this can be any value from the 
; IANA protocol registry, as per 
; [GRASP] section 2.9.5.1, note 3.
port-number      = selected by Proxy

Figure 6c: AN_Proxy CDDL

On a small network the Registrar MAY include the GRASP M_FLOOD announcements to locally connected networks.

The $transport-proto above indicates the method that the pledge-proxy-registrar will use. The TCP method described here is mandatory, and other proxy methods, such as CoAP methods not defined in this document are optional. Other methods MUST NOT be enabled unless the Join Registrar ASA indicates support for them in it’s own announcement.

4.2. CoAP connection to Registrar

The use of CoAP to connect from pledge to registrar is out of scope for this document, and is described in future work. See [I-D.ietf-anima-constrained-voucher].

4.3. Proxy discovery and communication of Registrar

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

The registrar announces itself using ACP instance of GRASP using M_FLOOD messages. ANI proxies MUST support GRASP discovery of registrars.

The M_FLOOD is formatted as follows:

[M_FLOOD, 12340815, h’fda379a6f6ee00000200000064000001’, 180000, 
["AN_join_registrar", 4, 255, "EST-TLS"], 
[O_IPv6_LOCATOR, 
h’fda379a6f6ee00000200000064000001’, IPPROTO_TCP, 80]]

Figure 7a: Registrar Discovery

The formal CDDL definition is:

flood-message = [M_FLOOD, session-id, initiator, ttl, 
+ [objective, (locator-option / [])]]

objective = ["AN_join_registrar", objective-flags, loop-count, 
objective-value]

initiator = ACP address to contact Registrar
objective-flags = sync-only ; as in GRASP spec
sync-only = 4 ; M_FLOOD only requires synchronization
loop-count = 255 ; mandatory maximum
objective-value = text ; name of the (list of) of supported
; protocols: "EST-TLS" for RFC7030.

Figure 7: AN_join_registrar CDDL

The M_FLOOD message MUST be sent periodically. The period is subject to network administrator policy (EST server configuration). It must be sufficiently low that the aggregate amount of periodic M_FLOODs from all EST servers causes negligible traffic across the ACP.
Here are some examples of locators for illustrative purposes. Only the first one ($\text{transport-protocol} = 6$, TCP) is defined in this document and is mandatory to implement.

\begin{verbatim}
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]
\end{verbatim}

A protocol of 6 indicates that TCP proxying on the indicated port is desired.

Registrars MUST announce the set of protocols that they support. They MUST support TCP traffic.

Registrars MUST accept HTTPS/EST traffic on the TCP ports indicated.

Registrars MUST support ANI TLS circuit proxy and therefore BRSKI across HTTPS/TLS native across the ACP.

In the ANI, the Autonomic Control Plane (ACP) secured instance of GRASP ([I-D.ietf-anima-grasp]) MUST be used for discovery of ANI registrar ACP addresses and ports by ANI proxies. The TCP leg of the proxy connection between ANI proxy and ANI registrar therefore also runs across the ACP.

5. Protocol Details (Pledge - Registrar - MASA)

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]. The goal of these extensions is 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://" iauthority "/.well-known/est".
BRSKI uses existing CMS message formats for existing EST operations. BRSKI 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 state occurs only once, and that the subsequent resolution of the provision state is not subject to a MITM attack during a critical phase.

Summarized automation extensions for the BRSKI-EST flow are:

- The pledge either attempts concurrent connections via each discovered proxy, or it times out quickly and tries connections in series, as explained at the end of Section 5.1.

- The pledge provisionally accepts the registrar certificate during the TLS handshake as detailed in Section 5.1.

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

- The pledge completes authentication of the server certificate as detailed in Section 5.6.1. This moves the BRSKI-EST TLS connection out of the provisional state.

- Mandatory bootstrap steps conclude with voucher status telemetry (see Section 5.7).

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 manufacturer trust anchor since serial numbers are not globally unique.

- The registrar requests and validates the voucher from the MASA.

- The registrar forwards the voucher to the pledge when requested.
o The registrar 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 registrar. 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" wherein the client is authenticated with the IDevID certificate, and the EST server (the registrar) is provisionally authenticated with an unverified server certificate.

The pledge maintains a security paranoia concerning the provisional state, and all data received, until a voucher is received and verified as specified in Section 5.6.1

A Pledge that can connect to multiple registries concurrently, SHOULD do so. Some devices may be unable to do so for lack of threading, or resource issues. Concurrent connections defeat attempts by a malicious proxy from causing a TCP Slowloris-like attack (see [slowloris]).

A pledge that can not maintain as many connections as there are eligible proxies. If no connection is making process after 5 seconds then the pledge SHOULD drop the oldest connection and go on to a different proxy: the proxy that has been communicated with least recently. If there were no other proxies discovered, the pledge MAY continue to wait, as long as it is concurrently listening for new proxy announcements.

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 "/.well-known/est/requestvoucher".

The pledge voucher-request Content-Type is:
The request is a "YANG-defined JSON document that has been signed using a CMS structure" as described in Section 3 using the JSON encoding described in [RFC7951]. This voucher media type is defined in [RFC8366] and is also used for the pledge voucher-request. The pledge SHOULD sign the request using the Section 2.3 credential.

Registrar implementations SHOULD anticipate future media types but of course will simply fail the request if those types are not yet known.

The pledge SHOULD include an [RFC7231] section 5.3.2 "Accept" header indicating the acceptable media type for the voucher response. The "application/voucher-cms+json" media type is defined in [RFC8366] but constrained voucher formats are expected in the future. Registrar's and MASA's are expected to be flexible in what they accept.

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 pledge voucher-request MUST contain a cryptographically strong random or pseudo-random number nonce. (see [RFC4086]) Doing so ensures Section 2.6.1 functionality. The nonce MUST NOT be reused for multiple bootstrapping attempts. (The registrar voucher-request MAY omit the nonce as per Section 3.1)

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 voucher-request if the "proximity" assertion is populated.

All other fields MAY be omitted in the pledge voucher-request.

An example JSON payload of a pledge voucher-request is in Section 3.3 Example 1.

The registrar validates the client identity as described in EST [RFC7030] section 3.3.2. The registrar confirms that the 'proximity' assertion and associated 'proximity-registrar-cert' are correct.

5.3. Registrar Authorization of Pledge

In a fully automated network all devices must be securely identified and authorized to join the domain.
A Registrar accepts or declines a request to join the domain, based on the authenticated identity presented. Automated acceptance criteria include:

- allow any device of a specific type (as determined by the X.509 IDevID),
- allow any device from a specific vendor (as determined by the X.509 IDevID),
- allow a specific device from a vendor (as determined by the X.509 IDevID) against a domain white list. (The mechanism for checking a shared white list potentially used by multiple Registrars is out of scope).

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 5.5) and returns that MASA signed voucher to the pledge as described in Section 5.6.

5.4. 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 2.8 for [RFC6125] authentication of the MASA.

The primary method of registrar "authentication" by the MASA is detailed in Section 5.5. As detailed in Section 11 the MASA might find it necessary to request additional registrar authentication.

The MASA and the registrars SHOULD be prepared to support TLS client certificate authentication and/or HTTP Basic or Digest authentication as described in [RFC7030] for EST clients. This connection MAY also have no client authentication at all (Section 7.4).

The authentication of the BRSKI-MASA connection does not affect the voucher-request process, as voucher-requests are already signed by the registrar. Instead, this authentication provides access control to the audit log.

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.

5.5. Registrar Requests Voucher from MASA

When a registrar receives a pledge voucher-request it in turn submits a registrar voucher-request to the MASA service via an HTTPS RESTful interface ([RFC7231]).

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

The voucher media type "application/voucher-cms+json" is defined in [RFC8366] and is also used for the registrar voucher-request. It is a JSON document that has been signed using a CMS structure. The registrar MUST sign the registrar voucher-request. The entire registrar certificate chain, up to and including the Domain CA, MUST be included in the CMS structure.

MASA implementations SHOULD anticipate future media types but of course will simply fail the request if those types are not yet known.

The Registrar SHOULD include an [RFC7231] section 5.3.2 "Accept" header indicating the response media types that are acceptable. This list SHOULD be the entire list presented to the Registrar in the Pledge’s original request (see Section 5.2) but MAY be a subset. MASA’s are expected to be flexible in what they accept.

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: This is the value from the pledge voucher-request. The registrar voucher-request MAY omit the nonce as per Section 3.1

serial-number: The serial number of the pledge the registrar would like a voucher for. The registrar determines this value by parsing the authenticated pledge IDevID certificate. See Section 2.3. The registrar MUST verify that the serial number field it parsed matches the serial number field the pledge provided in its voucher-request. This provides a sanity check useful for detecting error conditions and logging. The registrar MUST NOT simply copy the serial number field from a pledge voucher request as that field is claimed but not certified.

idevid-issuer: The idevid-issuer value from the pledge certificate is included to ensure a statistically unique identity.
prior-signed-voucher-request: The signed pledge voucher-request
SHOULD be included in the registrar voucher-request. (NOTE: what
is included is the complete pledge voucher-request, inclusive of
the ‘assertion’, ‘proximity-registrar-cert’, etc wrapped by the
pledge’s original signature). If a signed voucher-request was not
received from the pledge then this leaf is omitted from the
registrar voucher request.

A nonceless registrar voucher-request MAY be submitted to the MASA.
Doing so allows the registrar to request a voucher when the pledge is
deployed, or when the registrar anticipates not being able to connect
to the MASA while the pledge is being deployed. Some use cases
require the registrar to learn the appropriate IDevID SerialNumber
field and appropriate ‘Accept header’ field values from the physical
device labeling or from the sales channel (out-of-scope for this
document).

All other fields MAY be omitted in the registrar voucher-request.

Example JSON payloads of registrar voucher-requests are in
Section 3.3 Examples 2 through 4.

The MASA verifies that the registrar voucher-request is internally
consistent but does not necessarily authenticate the registrar
certificate since the registrar is not known to the MASA in advance.
The MASA performs the actions and validation checks described in the
following sub-sections before issuing a voucher.

5.5.1. MASA renewal of expired vouchers

As described in [RFC8366] vouchers are normally short lived to avoid
revocation issues. If the request is for a previous (expired)
voucher using the same registrar then the request for a renewed
voucher SHOULD be automatically authorized. The MASA has sufficient
information to determine this by examining the request, the registrar
authentication, and the existing audit log. The issuance of a
renewed voucher is logged as detailed in Section 5.6.

To inform the MASA that existing vouchers are not to be renewed one
can update or revoke the registrar credentials used to authorize the
request (see Section 5.5.3 and Section 5.5.4). More flexible methods
will likely involve sales channel integration and authorizations
(details are out-of-scope of this document).
5.5.2. MASA verification of voucher-request signature consistency

The MASA MUST verify that the registrar 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 registrar voucher-request. This verification is only a consistency check that the unauthenticated domain CA intended the voucher-request signer to be a registrar. Performing this check provides value to the domain PKI by assuring the domain administrator that the MASA service will only respect claims from authorized Registration Authorities of the domain.

The MASA verifies that the domain CA certificate is included in the CMS structure as detailed in Section 5.5.

5.5.3. MASA authentication of registrar (certificate)

If a nonceless voucher-request is submitted the MASA MUST authenticate the registrar as described in either EST [RFC7030] section 3.2, section 3.3, or by validating the registrar’s certificate used to sign the registrar voucher-request. Any of these methods reduce the risk of DDoS attacks and provide an authenticated identity as an input to sales channel integration and authorizations (details are out-of-scope of this document).

In the nonced case, validation of the registrar MAY be omitted if the device policy is to accept audit-only vouchers.

5.5.4. MASA revocation checking of registrar (certificate)

As noted in Section 5.5.3 the MASA performs registrar authentication in a subset of situations (e.g. nonceless voucher requests). Normal PKIX revocation checking is assumed during either EST client authentication or voucher-request signature validation. Similarly, as noted in Section 5.5.2, the MASA performs normal PKIX revocation checking during signature consistency checks (a signature by a registrar certificate that has been revoked is an inconsistency).

5.5.5. MASA verification of pledge prior-signed-voucher-request

The MASA MAY verify that the registrar voucher-request includes the 'prior-signed-voucher-request' field. If so the prior-signed-voucher-request MUST include a 'proximity-registrar-cert' that is consistent with the certificate used to sign the registrar voucher-request. Additionally the voucher-request serial-number leaf MUST match the pledge serial-number that the MASA extracts from the signing certificate of the prior-signed-voucher-request. The MASA is
aware of which pledges support signing of their voucher requests and can use this information to confirm proximity of the pledge with the registrar, thus ensuring that the BRSKI-EST TLS connection has no man-in-the-middle.

If these checks succeed the MASA updates the voucher and audit log assertion leafs with the "proximity" assertion.

5.5.6. MASA pinning of registrar

The registrar’s certificate chain is extracted from the signature method. The chain includes the domain CA certificate as specified in Section 5.5. This certificate is used to populate the "pinned-domain-cert" of the voucher being issued. The domainID (e.g., hash of the root public key) is determined from the pinned-domain-cert and is used to update the audit log.

5.5.7. MASA nonce handling

The MASA does not verify the nonce itself. If the registrar voucher-request contains a nonce, and the prior-signed-voucher-request is exist, then the MASA MUST verify that the nonce is consistent. (Recall from above that the voucher-request might not contain a nonce, see Section 5.5 and Section 5.5.3).

The MASA MUST use the nonce from the registrar voucher-request for the resulting voucher and audit log. The prior-signed-voucher-request nonce is ignored during this operation.

5.6. MASA and Registrar Voucher Response

The MASA voucher response to the registrar is forwarded without changes to the pledge; therefore this section applies to both the MASA and the registrar. The HTTP signaling described applies to both the MASA and registrar responses. A registrar either caches prior MASA responses or dynamically requests a new voucher based on local policy (it does not generate or sign a voucher). Registrar evaluation of the voucher itself is purely for transparency and audit purposes to further inform log verification (see Section 5.8.2) and therefore a registrar could accept future voucher formats that are opaque to the registrar.

If the voucher-request is successful, the server (MASA responding to registrar or registrar responding to pledge) 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. In this case, the response data from the MASA MUST be a plaintext human-
readable (ASCII, English) error message containing explanatory information describing why the request was rejected.

The registrar 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". (see [RFC7231] section 6.6.4) The pledge is RECOMMENDED to provide local feedback (blacked 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. Ideally the pledge would keep track of the appropriate Retry-After header values for any number of outstanding registrars but this would involve a state table on the pledge. Instead the pledge MAY ignore the exact Retry-After value in favor of a single hard coded value (a registrar that is unable to complete the transaction after the first 60 seconds has another chance a minute later). A pledge SHOULD only maintain a 202 retry-state for up to 4 days, which is longer than a long weekend, after which time the enrollment attempt fails and the pledge returns to discovery state.

In order to avoid infinite redirect loops, which a malicious registrar might do in order to keep the pledge from discovering the correct registrar, the pledge MUST NOT follow more than one redirection (3xx code) to another web origins. EST supports redirection but requires user input; this change allows the pledge to follow a single redirection without a user interaction.

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

A 404 (Not Found) response is appropriate when the request is for a device that 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) cannot be issued such as because the MASA knows the pledge cannot process that type. The registrar SHOULD use this response if it determines the pledge is unacceptable due to inventory control, MASA audit logs, or any other reason.

A 415 (Unsupported Media Type) response is appropriate for a request that has a voucher-request or accept encoding that is not understood.
The voucher response format is as indicated in the submitted accept header or based on the MASA's prior understanding of proper format for this Pledge. Only the [RFC8366] "application/voucher-cms+json" media type is defined at this time. The syntactic details of vouchers are described in detail in [RFC8366]. For example, the voucher consists of:

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

The MASA populates the voucher fields as follows:

nonce: The nonce from the pledge if available. See Section 5.5.7.

assertion: The method used to verify assertion. See Section 5.5.5.

pinned-domain-cert: The domain CA cert. See Section 5.5.6. This figure is illustrative, for an example, see Appendix D.2

serial-number: The serial-number as provided in the voucher-request. Also see Section 5.5.5.

domain-cert-revocation-checks: Set as appropriate for the pledge’s capabilities and as documented in [RFC8366]. The MASA MAY set this field to 'false' since setting it to 'true' would require that revocation information be available to the pledge and this document does not make normative requirements for [RFC6961] or equivalent integrations.

expires-on: This is set for nonceless vouchers. The MASA ensures the voucher lifetime is consistent with any revocation or pinned-domain-cert consistency checks the pledge might perform. See section Section 2.6.1. There are three times to consider: (a) a configured voucher lifetime in the MASA, (b) the expiry time for the registrar’s certificate, (c) any certificate revocation information (CRL) lifetime. The expires-on field SHOULD be before the earliest of these three values. Typically (b) will be some significant time in the future, but (c) will typically be short (on the order of a week or less). The RECOMMENDED period for (a) is on the order of 20 minutes, so it will typically determine the lifespan of the resulting voucher. 20 minutes is sufficient time to reach the post-provisional state in the pledge, at which point
there is an established trust relationship between pledge and registrar. The subsequent operations can take as long as required from that point onwards. The lifetime of the voucher has no impact on the lifespan of the ownership relationship.

Whenever a voucher is issued the MASA MUST update the audit log appropriately. The internal state requirements to maintain the audit log are out-of-scope. See Section 5.8.1 for a discussion of reporting the log to a registrar.

5.6.1. Pledge voucher verification

The pledge MUST verify the voucher signature using the manufacturer installed trust anchor(s) associated with the manufacturer’s MASA (this is likely included in the pledge’s firmware). Management of the manufacturer installed trust anchor(s) is out-of-scope of this document; this protocol does not update these trust anchor(s).

The pledge MUST verify the serial-number field of the signed voucher matches the pledge’s own serial-number.

The pledge MUST verify that the voucher nonce field is accurate and matches the nonce the pledge submitted to this registrar, or that the voucher is nonceless (see Section 7.2).

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 that it does not recognize.

5.6.2. Pledge authentication of provisional TLS connection

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.

If a registrar’s credentials cannot 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 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 2.6.1. Once
the PKIX path validation is successful the TLS connection is no
longer provisional.

The pinned-domain-cert MAY be installed as an trust anchor for future
operations such as enrollment (e.g. [RFC7030] as recommended) or
trust anchor management or raw protocols that do not need full PKI
based key management. It 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 [RFC7030] 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.

5.7. Pledge BRISKI 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 may be 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 registrar expected to continue
joining the domain.
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 JSON fields in this POST MAY be added, see Section 8.3.

5.8. Registrar audit log request

After receiving the pledge status telemetry Section 5.7, the registrar SHOULD request the MASA audit log from the MASA service.

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

The registrar SHOULD HTTP POST the same registrar voucher-request as it did when requesting a voucher (using the same Content-Type). It is posted to the /requestauditlog URI instead. The "idevid-issuer" and "serial-number" informs the MASA 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 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.

A registrar MAY request logs at future times. If the registrar generates a new request then the MASA is forced to perform the additional cryptographic operations to verify the new request.

A MASA that receives a request for a device that does not exist, or for which the requesting owner was never an owner returns an HTTP 404 ("Not found") code.

Rather than returning the audit log as a response to the POST (with a return code 200), the MASA MAY instead return a 201 ("Created")
RESTful response ([RFC7231] section 7.1) containing a URL to the prepared (and easily cachable) audit response.

In order to avoid enumeration of device audit logs, MASA that return URLs SHOULD take care to make the returned URL unguessable. For instance, rather than returning URLs containing a database number such as https://example.com/auditlog/1234 or the EUI of the device such https://example.com/auditlog/10-00-00-11-22-33, the MASA SHOULD return a randomly generated value (a "slug" in web parlance). The value is used to find the relevant database entry.

A MASA that returns a code 200 MAY also include a Location: header for future reference by the registrar.

5.8.1. MASA audit log response

A log data file is returned consisting of all log entries associated with the the device selected by the IDevID presented in the request. The audit log may be truncated of old or repeated values as explained below. The returned data is in JSON format ([RFC7951]), and the Content-Type SHOULD be "application/json". For example:

```json
{
    "version": "1",
    "events": [
        {
            "date": "<date/time of the entry>",
            "domainID": "<domainID extracted from voucher-request>",
            "nonce": "<any nonce if supplied (or the exact string 'NULL')>",
            "assertion": "<the value from the voucher assertion leaf>",
            "truncated": "<the number of domainID entries truncated>",
        },
        {
            "date": "<date/time of the entry>",
            "domainID": "<anotherDomainID extracted from voucher-request>",
            "nonce": "<any nonce if supplied (or the exact string 'NULL')>",
            "assertion": "<the value from the voucher assertion leaf>",
        }
    ],
    "truncation": {
        "nonced duplicates": "<total number of entries truncated>",
        "nonceless duplicates": "<total number of entries truncated>",
        "arbitrary": "<number of domainID entries removed entirely>",
    }
}
```

Distribution of a large log is less than ideal. This structure can be optimized as follows: Nonced or Nonceless entries for the same
domainID MAY be truncated from the log leaving only the single most recent nonced or nonceless entry for that domainID. In the case of truncation the ‘event’ truncation value SHOULD contain a count of the number of events for this domainID that were truncated. The log SHOULD NOT be further reduced but there could exist operational situation where maintaining the full log is not possible. In such situations the log MAY be arbitrarily truncated for length, with the number of removed entries indicated as ‘arbitrary’.

If the truncation count exceeds 1024 then the MASA MAY use this value without further incrementing it.

A log where duplicate entries for the same domain have been truncated ("nonced duplicates" and/or "nonceless duplicates) could still be acceptable for informed decisions. A log that has had "arbitrary" truncations is less acceptable but manufacturer transparency is better than hidden truncations.

This document specifies a simple log format as provided by the MASA service to the registrar. This format could be improved by distributed consensus technologies that integrate vouchers with technologies such as block-chain or hash trees or optimized logging approaches. Doing so is out of the scope of this document but is an anticipated improvement 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.

5.8.2. Registrar audit log verification

Each time the Manufacturer Authorized Signing Authority (MASA) issues a voucher, it places it into the audit log for that device. The details are described in Section 5.8. The contents of the audit log can express a variety of trust levels, and this section explains what kind of trust a registrar can derive from the entries.

While the audit log provides a list of vouchers that were issued by the MASA, the vouchers are issued in response to voucher-requests, and it is the contents of the voucher-requests which determines how meaningful the audit log entries are.

A registrar SHOULD use the log information to make an informed decision regarding the continued bootstrapping of the pledge. The exact policy is out of scope of this document as it depends on the security requirements within the registrar domain. Equipment that is purchased pre-owned can be expected to have an extensive history. The following discussion is provided to help explain the value of each log element:
date: The date field provides the registrar an opportunity to divide the log around known events such as the purchase date. Depending on context known to the registrar or administrator events before/after certain dates can have different levels of importance. For example for equipment that is expected to be new, and thus have no history, it would be a surprise to find prior entries.

domainID: If the log includes an unexpected domainID then the pledge could have imprinted on an unexpected domain. The registrar can be expected to use a variety of techniques to define "unexpected" ranging from white lists of prior domains to anomaly detection (e.g. "this device was previously bound to a different domain than any other device deployed"). Log entries can also be compared against local history logs in search of discrepancies (e.g. "this device was re-deployed some number of times internally but the external audit log shows additional re-deployments our internal logs are unaware of").

nonce: Noneceless entries mean the logged domainID could theoretically trigger a reset of the pledge and then take over management by using the existing nonceless voucher.

assertion: The assertion leaf in the voucher and audit log indicates why the MASA issued the voucher. A "verified" entry means that the MASA issued the associated voucher as a result of positive verification of ownership but this can still be problematic for registrar's that expected only new (not pre-owned) pledges. A "logged" assertion informs the registrar that the prior vouchers were issued with minimal verification. A "proximity" assertion assures the registrar that the pledge was truly communicating with the prior domain and thus provides assurance that the prior domain really has deployed the pledge.

A relatively simple policy is to white list known (internal or external) domainIDs and to require all vouchers to have a nonce and/or require that all nonceless vouchers be from a subset (e.g. only internal) domainIDs. A simple action is to revoke any locally issued credentials for the pledge in question or to refuse to forward the voucher. 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.

5.9. 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.

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.

5.9.1. EST Distribution of CA Certificates

The pledge SHOULD 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 5.6.1 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.

5.9.2. EST CSR Attributes

Automated bootstrapping occurs without local administrative configuration of the pledge. In some deployments it is 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 bootstrapping 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 registrar MUST also confirm that the resulting CSR is formatted as indicated before forwarding the request to a CA. If the registrar is communicating with the CA using a protocol such as full CMC, which provides mechanisms to override the CSR attributes, then these mechanisms MAY be used even if the client ignores CSR Attribute guidance.

5.9.3. EST Client Certificate Request

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

5.9.4. Enrollment Status Telemetry

For automated bootstrapping of devices, the administrative 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. [RFC7030] 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.

```json
{
  "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.

5.9.5. Multiple certificates

Pledges that require multiple certificates could establish direct EST
connections to the registrar.

5.9.6. 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 definition 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.ietf-ace-coap-est] and that CoAP
mappings for BRSKI will be discussed either there or in future work.
6. Clarification of transfer-encoding

[RFC7030] defines its endpoints to include a "Content-Transfer-Encoding" heading, and the payloads to be [RFC4648] Base64 encoded DER.

When used within BRSKI, the original RFC7030 EST endpoints remain Base64 encoded, but the new BRSKI endpoints which send and receive binary artifacts (specifically, ../voucherrequest) are binary. That is, no encoding is used.

In the BRSKI context, the EST "Content-Transfer-Encoding" header if present, SHOULD be ignored. This header does not need to be included.

7. 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.

This section is considered non-normative: use suggested methods MUST be detailed in specific profiles of BRSKI. This is the subject for future work.

7.1. Trust Model

This section explains the trust relationships detailed in Section 2.4:

```
+--------+         +---------+    +------------+     +------------+
| Pledge |         | Join    |    | Domain     |     | Manufacturer|
|        |         | Proxy   |    | Registrar  |     | Service     |
|        |         |         |    |            |     | (Internet) |
+--------+         +---------+    +------------+     +------------+
```

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.

Join Proxy: Provides proxy functionalities but is not involved in security considerations.
Registrar: When interacting with a MASA a registrar makes all decisions. For Ownership Audit Vouchers (see [RFC8366]) the registrar is provided an opportunity to accept MASA decisions.

Vendor Service, MASA: This form of manufacturer 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 manufacturer.

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

7.2. 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 a use case where the registrar can not connect to the MASA at the deployment time. Logging and validity periods address the security considerations of supporting these use cases.

2. Many devices already support "trust on first use" for physical interfaces such as console ports. This document does not change that reality. Devices supporting this protocol MUST NOT support "trust on first use" on network interfaces. This is because "trust on first use" over network interfaces would undermine the logging based security protections provided by this specification.

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 manufacturer service is unavailable. This behavior SHOULD be available via local configuration or physical presence methods (such as use of a serial/craft console) 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 any method of skipping voucher validation (including use of craft serial console) only be available if hardware assisted Network Endpoint Assessment [RFC5209] is supported. This recommendation ensures that domain network

monitoring can detect inappropriate use of offline or emergency deployment procedures when voucher-based bootstrapping is not used.

7.3. 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 5.5 format to ensure the pledge’s serial number information is provided to the registrar (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 submit a nonceless voucher-requests to the MASA service (by not including a nonce in the voucher-request.) The resulting vouchers can then be stored by the registrar until they are needed during bootstrapping operations. This is for use cases where the target network is protected by an air gap and therefore cannot contact the MASA service during pledge deployment.

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

5. A registrar MAY accept voucher formats of future types that can not be parsed by the Registrar. This reduces the Registrar’s
visibility into the exact voucher contents but does not change the protocol operations.

7.4. MASA security reductions

Lower security modes chosen by the MASA service affect 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 a 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 a 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 and authorized to provide this functionality. The MASA 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 a voucher. This is expected to be a common operational model because doing so relieves the manufacturer 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-request' information for pledges that support that functionality. This provides a proof-of-proximity check that reduces the need for ownership verification.

8. IANA Considerations

This document requires the following IANA actions:
8.1. Well-known EST registration

This document extends the definitions of "est" (so far defined via
RFC7030) in the "https://www.iana.org/assignments/well-known-uris/
well-known-uris.xhtml" registry as follows:

- add /.well-known/est/requestvoucher (see Section 5.5)
- add /.well-known/est/requestauditlog (see Section 5.7)

8.2. 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.

This document has received an early allocation from the id-pe
registry (SMI Security for PKIX Certificate Extension) for id-pe-
masa-url with the value 32, resulting in an OID of
1.3.6.1.5.5.7.1.32.

8.3. Pledge BRSKI Status Telemetry

IANA is requested to create a new Registry entitled: "BRSKI
Parameters", and within that Registry to create a table called:
"Pledge BRSKI 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 (Section 5.7) as the
reference:

- version
- Status
- Reason
- reason-context

8.4. DNS Service Names

IANA is requested to register the following Service Names:
Service Name: _brski-proxy
Transport Protocol(s): tcp
Assignee: IESG <iesg@ietf.org>.
Contact: IESG <iesg@ietf.org>
Description: The Bootstrapping Remote Secure Key
Infrastructures Proxy
Reference: [This document]

Service Name: _brski-registrar
Transport Protocol(s): tcp
Assignee: IESG <iesg@ietf.org>.
Contact: IESG <iesg@ietf.org>
Description: The Bootstrapping Remote Secure Key
Infrastructures Registrar
Reference: [This document]

8.5. MUD File Extension for the MASA

The IANA is requested to list the name "masa" in the MUD extensions registry defined in [I-D.ietf-opsawg-mud]. Its use is documented in Appendix C.

9. Applicability to the Autonomic Control Plane

This document provides a solution to the requirements for secure bootstrap set out in Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance [RFC8368], A Reference Model for Autonomic Networking [I-D.ietf-anima-reference-model] and specifically the An Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane], section 3.2 (Secure Bootstrap), and section 6.1 (ACP Domain, Certificate and Network).

The protocol described in this document has appeal in a number of other non-ANIMA use cases. Such uses of the protocol will be deploying into other environments with different tradeoffs of privacy, security, reliability and autonomy from manufacturers. As such those use cases will need to provide their own applicability statements, and will need to address unique privacy and security considerations for the environments in which they are used.

The autonomic control plane that this document provides bootstrap for is typically a medium to large Internet Service Provider organization, or an equivalent Enterprise that has significant layer-3 router connectivity. (A network consisting of primarily layer-2 is not excluded, but the adjacencies that the ACP will create and maintain will not reflect the topology until all devices participate in the ACP).
As specified in the ANIMA charter, this work "..focuses on professionally-managed networks." Such a network has an operator and can do things like install, configure and operate the Registrar function. The operator makes purchasing decisions and is aware of what manufacturers it expects to see on its network.

Such an operator also is capable of performing the traditional (craft serial-console) based bootstrap of devices. The zero-touch mechanism presented in this and the ACP document represents a significant efficiency: in particular it reduces the need to put senior experts on airplanes to configure devices in person. There is a recognition as the technology evolves that not every situation may work out, and occasionally a human still still have to visit.

The BRSKI protocol is going into environments where there have already been quite a number of vendor proprietary management systems. Those are not expected to go away quickly, but rather to leverage the secure credentials that are provisioned by BRSKI. The connectivity requirements of said management systems are provided by the ACP.

10. Privacy Considerations

10.1. MASA audit log

The MASA audit log includes a hash of the domainID for each Registrar a voucher has been issued to. This information is closely related to the actual domain identity, especially when paired with the anti-DDoS authentication information the MASA might collect. This could provide sufficient information for the MASA service to build a detailed understanding the devices that have been provisioned within a domain.

There are a number of design choices that mitigate this risk. The domain can maintain some privacy since it has not necessarily been authenticated and is not authoritatively bound to the supply chain.

Additionally the domainID captures only the unauthenticated subject key identifier of the domain. A privacy sensitive domain could theoretically generate a new domainID for each device being deployed. Similarly a privacy sensitive domain would likely purchase devices that support proximity assertions from a manufacturer that does not require sales channel integrations. This would result in a significant level of privacy while maintaining the security characteristics provided by Registrar based audit log inspection.
10.2. What BRSKI-MASA reveals to the manufacturer

The so-called "call-home" mechanism that occurs as part of the BRSKI-MASA connection standardizes what has been deemed by some as a sinister mechanism for corporate oversight of individuals. ([livingwithIoT] and [IoTstrangeThings] for a small sample).

As the Autonomic Control Plane (ACP) usage of BRSKI is not targeted at individual usage of IoT devices, but rather at the Enterprise and ISP creation of networks in a zero-touch fashion, the "call-home" represents a different kind of concern.

It needs to be re-iterated that the BRSKI-MASA mechanism only occurs once during the commissioning of the device. It is well defined, and although encrypted with TLS, it could in theory be made auditable as the contents are well defined. This connection does not occur when the device powers on or is restarted for normal routines. It is conceivable that a device could be forced to go through a full factory reset during an exceptional firmware update situation, after which enrollment would have be repeated.

The BRSKI call-home mechanism is mediated via the owner’s Registrar, and the information that is transmitted is directly auditable by the device owner. This is in stark contrast to many "call-home" protocols where the device autonomously calls home and uses an undocumented protocol.

While the contents of the signed part of the pledge voucher request can not be changed, they are not encrypted at the registrar. The ability to audit the messages by the owner of the network prevents exfiltration of data by a nefarious pledge. The contents of an unsigned voucher request are, however, completely changeable by the Registrar. Both are, to re-iterate, encrypted by TLS while in transit.

The BRSKI-MASA exchange reveals the following information to the manufacturer:

- the identity of the device being enrolled (down to the serial-number!).
- an identity of the domain owner in the form of the domain trust anchor. However, this is not a global PKI anchored name within the WebPKI, so this identity could be pseudonymous. If there is sales channel integration, then the MASA will have authenticated the domain owner, either via pinned certificate, or perhaps another HTTP authentication method, as per Section 5.5.3.
the time the device is activated,

- the IP address of the domain Owner’s Registrar. For ISPs and Enterprises, the IP address provides very clear geolocation of the owner. No amount of IP address privacy extensions ([RFC4941]) can do anything about this, as a simple whois lookup likely identifies the ISP or Enterprise from the upper bits anyway. A passive attacker who observes the connection definitely may conclude that the given enterprise/ISP is a customer of the particular equipment vendor. The precise model that is being enrolled will remain private.

The above situation is to be distinguished from a residential/individual person who registers a device from a manufacturer: that an enterprise/ISP purchases routing products is hardly worth mentioning. Deviations would, however, be notable.

The situation is not improved by the enterprise/ISP using anonymization services such as Tor [Dingledine2004], as a TLS 1.2 connection will reveal the ClientCertificate used, clearly identifying the enterprise/ISP involved. TLS 1.3 is better in this regard, but an active attacker can still discover the parties involved by performing a Man-In-The-Middle-Attack on the first attempt (breaking/killing it with a TCP RST), and then letting subsequent connection pass through.

A manufacturer could attempt to mix the BRSKI-MASA traffic in with general traffic their site by hosting the MASA behind the same (set) of load balancers that the companies normal marketing site is hosted behind. This makes lots of sense from a straight capacity planning point of view as the same set of services (and the same set of Distributed Denial of Service mitigations) may be used. Unfortunately, as the BRSKI-MASA connections include TLS ClientCertificate exchanges, this may easily be observed in TLS 1.2, and a traffic analysis may reveal it even in TLS 1.3. This does not make such a plan irrelevant. There may be other organizational reasons to keep the marketing site (which is often subject to frequent redesigs, outsourcing, etc.) separate from the MASA, which may need to operate reliably for decades.

10.3. Manufacturers and Used or Stolen Equipment

As explained above, the manufacturer receives information each time that a device which is in factory-default mode does a zero-touch bootstrap, and attempts to enroll into a domain owner’s registrar.
The manufacturer is therefore in a position to decline to issue a voucher if it detects that the new owner is not the same as the previous owner.

1. This can be seen as a feature if the equipment is believed to have been stolen. If the legitimate owner notifies the manufacturer of the theft, then when the new owner brings the device up, if they use the zero-touch mechanism, the new (illegitimate) owner reveals their location and identity.

2. In the case of Used equipment, the initial owner could inform the manufacturer of the sale, or the manufacturer may just permit resales unless told otherwise. In which case, the transfer of ownership simply occurs.

3. A manufacturer could however decide not to issue a new voucher in response to a transfer of ownership. This is essentially the same as the stolen case, with the manufacturer having decided that the sale was not legitimate.

4. There is a fourth case, if the manufacturer is providing protection against stolen devices. The manufacturer then has a responsibility to protect the legitimate owner against fraudulent claims that the equipment was stolen. Such a claim would cause the manufacturer to refuse to issue a new voucher. Should the device go through a deep factory reset (for instance, replacement of a damaged main board component, the device would not bootstrap.

5. Finally, there is a fifth case: the manufacturer has decided to end-of-line the device, or the owner has not paid a yearly support amount, and the manufacturer refuses to issue new vouchers at that point. This last case is not new to the industry: many license systems are already deployed that have significantly worse effect.

This section has outlined five situations in which a manufacturer could use the voucher system to enforce what are clearly license terms. A manufacturer that attempted to enforce license terms via vouchers would find it rather ineffective as the terms would only be enforced when the device is enrolled, and this is not (to repeat), a daily or even monthly occurrence.

10.4. Manufacturers and Grey market equipment

Manufacturers of devices often sell different products into different regional markets. Which product is available in which market can be driven by price differentials, support issues (some markets may

require manuals and tech-support to be done in the local language),
government export regulation (such as whether strong crypto is
permitted to be exported, or permitted to be used in a particular
market). When an domain owner obtains a device from a different
market (they can be new) and transfers it to a different location,
this is called a Grey Market.

A manufacturer could decide not to issue a voucher to an enterprise/
ISP based upon their location. There are a number of ways which this
could be determined: from the geolocation of the registrar, from
sales channel knowledge about the customer, and what products are
(un)-available in that market. If the device has a GPS the
coordinates of the device could even be placed into an extension of
the voucher.

The above actions are not illegal, and not new. Many manufacturers
have shipped crypto-weak (exportable) versions of firmware as the
default on equipment for decades. The first task of an enterprise/
ISP has always been to login to a manufacturer system, show one’s
"entitlement" (country information, proof that support payments have
been made), and receive either a new updated firmware, or a license
key that will activate the correct firmware.

BRSKI permits the above process to automated (in an autonomic
fashion), and therefore perhaps encourages this kind of
differentiation by reducing the cost of doing it.

An issue that manufacturers will need to deal with in the above
automated process is when a device is shipped to one country with one
set of rules (or laws or entitlements), but the domain registry is in
another one. Which rules apply is something will have to be worked
out: the manufacturer could come to believe they are dealing with
Grey market equipment, when it is simply dealing with a global
enterprise.

10.5. Some mitigations for meddling by manufacturers

The most obvious mitigation is not to buy the product. Pick
manufacturers that are up-front about their policies, who do not
change them gratuitously.

A manufacturer could provide a mechanism to manage the trust anchors
and built-in certificates (IDevID) as an extension. This is a
substantial amount of work, and may be an area for future
standardization work.

Replacement of the voucher validation anchors (usually pointing to
the original manufacturer’s MASA) with those of the new owner permits
the new owner to issue vouchers to subsequent owners. This would be
done by having the selling (old) owner to run a MASA.

In order to automatically find the new MASA, the mechanism describe
in this document is to look for the MASA URL extension in the IDevID.
A new owner could override this in their Registrar, or the
manufacturer could provide a mechanism to update or replace the
IDevID prior to sale.

Once the voucher trust anchor and the IDevID is replaced, then the
device will no longer trust the manufacturer in any way. When a new
owner performs a bootstrap, the device will point to a MASA that has
been chosen, and will validate vouchers from this new entity.

The BRSKI protocol depends upon a trust anchor on the device and an
identity on the device. Management of these these entities
facilitiates a few new operational modes without making any changes to
the BRSKI protocol. Those modes include: offline modes where the
domain owner operates an internal MASA for all devices, resell modes
where the first domain owner becomes the MASA for the next (resold-
to) domain owner, and services where an aggregator acquires a large
variety of devices, and then acts as a pseudonymized MASA for a
variety of devices from a variety of manufacturers.

Some manufacturers may wish to consider replacement of the IDevID as
an indication that the device’s warantee is terminated. For others,
the privacy requirments of some deployments might consider this a
standard operating practice.

As discussed at the end of Section 5.8.1, new work could be done to
use a distributed consensus technology for the audit log. This would
permit the audit log to continue to be useful, even when there is a
chain of MASA due to changes of ownership.

11. Security Considerations

This document details a protocol for bootstrapping that balances
operational concerns against security concerns. As detailed in the
introduction, and touched on again in Section 7, the protocol allows
for reduced security modes. These attempt to deliver additional
control to the local administrator and owner in cases where less
security provides operational benefits. This section goes into more
detail about a variety of specific considerations.

To facilitate logging and administrative oversight, in addition to
triggering Registration verification of MASA logs, the pledge reports
on voucher parsing status to the registrar. In the case of a
failure, this information is informative to a potentially malicious
registrar. This is mandated anyway because of the operational benefits of an informed administrator in cases where the failure is indicative of a problem. The registrar is RECOMMENDED to verify MASA logs if voucher status telemetry is not received.

To facilitate truly limited clients EST RFC7030 section 3.3.2 requirements that the client MUST support a client authentication model have been reduced in Section 7 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 the pledge MUST treat all HTTP header and content data as untrusted data. HTTP libraries are regularly exposed to non-secured HTTP traffic: mature libraries should not have any problems.

Pledges 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 multiple vouchers 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 registrars verifying log information will see multiple entries and take this into account for their analytics purposes.

11.1. DoS against MASA

There are uses cases where the MASA could be unavailable or uncooperative to the Registrar. They include active DoS attacks, 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 in that MASA behavior might limit the ability to bootstrap 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 bootstrap their devices.

The issuance of nonceless vouchers themselves creates 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 and by Pledges only bootstrapping when in a factory default state. This reflects a balance between enabling MASA independence during future bootstrapping and the security of
bootstrapping itself. Registrar control over requesting and auditing nonceless vouchers allows device owners to choose an appropriate balance.

The MASA is exposed to DoS attacks wherein attackers claim an unbounded number of devices. Ensuring a registrar is representative of a valid manufacturer customer, even without validating ownership of specific pledge devices, helps to mitigate this. Pledge signatures on the pledge voucher-request, as forwarded by the registrar in the prior-signed-voucher-request field of the registrar voucher-request, 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. Supply chain integration ("know your customer") is an additional step that MASA providers and device vendors can explore.

11.2. Freshness in Voucher-Requests

A concern has been raised that the pledge voucher-request should contain some content (a nonce) provided by the registrar and/or MASA in order for those actors to verify that the pledge 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 pledge voucher-request.

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 registrar 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 registrar verifying the audit logs available from the MASA as described in Section 5.8. Rm might chose to collect a voucher-request but wait until after the intended registrar completes the authorization process before submitting it. This pledge voucher-request would be ‘stale’ in that it has a nonce that no longer matches the internal state of the pledge. In order to successfully use any resulting voucher the Rm would need to remove the stale nonce or anticipate the pledge’s future nonce state. Reducing the possibility of this is why the pledge is mandated to generate a strong random or pseudo-random number nonce.

Additionally, in order to successfully use the resulting voucher the 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.
- Retrieval and examination of MASA log information upon the occurrence of any such unexpected events. Rm will be listed in the logs along with nonce information for analysis.

11.3. Trusting manufacturers

The BRSKI extensions to EST permit a new pledge to be completely configured with domain specific trust anchors. The link from built-in manufacturer-provided trust anchors to domain-specific trust anchors is mediated by the signed voucher artifact.

If the manufacturer’s IDevID signing key is not properly validated, then there is a risk that the network will accept a pledge that should not be a member of the network. As the address of the manufacturer’s MASA is provided in the IDevID using the extension from Section 2.3, the malicious pledge will have no problem collaborating with it’s MASA to produce a completely valid voucher.

BRSKI does not, however, fundamentally change the trust model from domain owner to manufacturer. Assuming that the pledge used its
IDevID with RFC7030 EST and BRSKI, the domain (registrar) still needs to trust the manufacturer.

Establishing this trust between domain and manufacturer is outside the scope of BRSKI. There are a number of mechanisms that can adopted including:

- Manually configuring each manufacturer’s trust anchor.
- A Trust-On-First-Use (TOFU) mechanism. A human would be queried upon seeing a manufacturer’s trust anchor for the first time, and then the trust anchor would be installed to the trusted store. There are risks with this; even if the key to name is validated using something like the WebPKI, there remains the possibility that the name is a look alike: e.g, dem0.example. vs demO.example.
- scanning the trust anchor from a QR code that came with the packaging (this is really a manual TOFU mechanism)
- some sales integration process where trust anchors are provided as part of the sales process, probably included in a digital packing "slip", or a sales invoice.
- consortium membership, where all manufacturers of a particular device category (e.g, a light bulb, or a cable-modem) are signed by an certificate authority specifically for this. This is done by CableLabs today. It is used for authentication and authorization as part of TR-79: [docsisroot] and [TR069].

The existing WebPKI provides a reasonable anchor between manufacturer name and public key. It authenticates the key. It does not provide a reasonable authorization for the manufacturer, so it is not directly useable on it’s own.

11.4. Manufacturer Maintainance of trust anchors

BRSKI depends upon the manufacturer building in trust anchors to the pledge device. The voucher artifact which is signed by the MASA will be validated by the pledge using that anchor. This implies that the manufacturer needs to maintain access to a signing key that the pledge can validate.

The manufacturer will need to maintain the ability to make signatures that can be validated for the lifetime that the device could be onboarded. Whether this onboarding lifetime is less than the device lifetime depends upon how the device is used. An inventory of devices kept in a warehouse as spares might not be onboarded for many decades.
There are good cryptographic hygiene reasons why a manufacturer would not want to maintain access to a private key for many decades. A manufacturer in that situation can leverage a long-term certificate authority anchor, built-in to the pledge, and then a certificate chain may be incorporated using the normal CMS certificate set. This may increase the size of the voucher artifacts, but that is not a significant issue in non-constrained environments.

There are a few other operational variations that manufacturers could consider. For instance, there is no reason that every device need have the same set of trust anchors pre-installed. Devices built in different factories, or on different days, or any other consideration could have different trust anchors built in, and the record of which batch the device is in would be recorded in the asset database. The manufacturer would then know which anchor to sign an artifact against.

Aside from the concern about long-term access to private keys, a major limiting factor for the shelf-life of many devices will be the age of the cryptographic algorithms included. A device produced in 2019 will have hardware and software capable of validating algorithms common in 2019, and will have no defense against attacks (both quantum and von-neuman brute force attacks) which have not yet been invented. This concern is orthogonal to the concern about access to private keys, but this concern likely dominates and limits the lifespan of a device in a warehouse. If any update to firmware to support new cryptographic mechanism were possible (while the device was in a warehouse), updates to trust anchors would also be done at the same time.

12. Acknowledgements

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

Significant reviews were done by Jari Arko, Christian Huitema and Russ Housley.

13. References

13.1. Normative References

[I-D.ietf-anima-autonomic-control-plane]


13.2. Informative References

[Dingledine2004]
Dingledine, R., Mathewson, N., and P. Syverson, "Tor: the second-generation onion router", 2004,
https://spec.torproject.org/tor-spec/.

[docsisroot]
"CableLabs Digital Certificate Issuance Service", February 2018,

[I-D.ietf-ace-coap-est]
Stok, P., Kampanakis, P., Richardson, M., and S. Raza,
"EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-est-12 (work in progress), June 2019.

[I-D.ietf-anima-constrained-voucher]
Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols",
draft-ietf-anima-constrained-voucher-03 (work in progress), March 2019.
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L.,
and J. Nobre, "A Reference Model for Autonomic
Networking", draft-ietf-anima-reference-model-10 (work in
progress), November 2018.

Eckert, T. and M. Behringer, "Using Autonomic Control
Plane for Stable Connectivity of Network OAM", draft-ietf-
anima-stable-connectivity-10 (work in progress), February
2018.

Birkholz, H., Vigano, C., and C. Bormann, "Concise data
definition language (CDDL): a notational convention to
express CBOR and JSON data structures", draft-ietf-cbor-
cddl-08 (work in progress), March 2019.

Touch Provisioning (SZTP)", draft-ietf-netconf-
zerotouch-29 (work in progress), January 2019.

Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage
Description Specification", draft-ietf-opsawg-mud-25 (work
in progress), June 2018.

Richardson, M., "Considerations for stateful vs stateless
join router in ANIMA bootstrap", draft-richardson-anima-
state-for-joinrouter-02 (work in progress), January 2018.

"Wikipedia article: Imprinting", July 2015,

"IoT of toys stranger than fiction: Cybersecurity and data
privacy update (accessed 2018-12-02)", March 2017,
<https://www.welivesecurity.com/2017/03/03/

"What is it actually like to live in a house filled with
IoT devices? (accessed 2018-12-02)", February 2018,
<https://www.siliconrepublic.com/machines/
iot-smart-devices-reality>.


Appendix A. IPv4 and non-ANI operations

The specification of BRSKI in Section 4 intentionally only covers the mechanisms for an IPv6 pledge using Link-Local addresses. This section describes non-normative extensions that can be used in other environments.

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 Link-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

Pledge discovery of the proxy (Section 4.1) MAY be performed with DNS-based Service Discovery [RFC6763] over Multicast DNS [RFC6762] to discover the proxy at "_brski-proxy._tcp.local.".

Proxy discovery of the registrar (Section 4.3) MAY be performed with DNS-based Service Discovery over Multicast DNS to discover registrars by searching for the service "_brski-registrar._tcp.local.".

To prevent unacceptable levels of network traffic, when using mDNS, 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.

Discovery of registrar MAY also be performed with DNS-based service discovery by searching for the service "_brski-registrar._tcp.example.com". In this case the domain "example.com" is discovered as described in [RFC6763] section 11 (Appendix A.2 suggests the use of DHCP parameters).

If no local proxy or registrar service is located using the GRASP mechanisms or the above mentioned DNS-based Service Discovery methods the pledge MAY contact a well known manufacturer provided bootstrapping server by performing a DNS lookup using a well known URI such as "brski-registrar.manufacturer.example.com". The details of the URI are manufacturer specific. Manufacturers that leverage this method on the pledge are responsible for providing the registrar service. Also see Section 2.7.

The current DNS services returned during each query are 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 _bootstraps._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. 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> file "ietf-mud-extension@2018-02-14.yang"
module ietf-mud-brski-masa {
  yang-version 1.1;
  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 2018-02-14 {
    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>

The MUD extensions string "masa" is defined, and MUST be included in the extensions array of the mud container of a MUD file when this extension is used.
Appendix D. Example Vouchers

Three entities are involved in a voucher: the MASA issues (signs) it, the registrar’s public key is mentioned in the voucher, and the pledge validates it. In order to provide reproducible examples the public and private keys for an example MASA and registrar are first listed.

D.1. Keys involved

The Manufacturer has a Certificate Authority that signs the pledge’s IDevID. In addition the Manufacturer’s signing authority (the MASA) signs the vouchers, and that certificate must distributed to the devices at manufacturing time so that vouchers can be validated.

D.1.1. MASA key pair for voucher signatures

This private key signs vouchers:

```
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAgirQyKoEcfOfyRvmZ5P5Azn58tu7nS6y7oGFnCeINo+BmbgRho
r61cU60gwVagwYFk4EEACHhZANiAATZAH3Rb2FvJIOnts+vxuWW35ofyNbcHzzjA
zoI2kWFZFE1ByurKI+McNMFGiGrGnRXIGqWCfw5ICgJ8CuM3v5ty9bf7KU0kejz
Tv+5PV++elkP9H9Q83vqTaws2WwWTxI=
-----END EC PRIVATE KEY-----
```

This public key validates vouchers:

```
-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQDAjBNNMRIwEAYKzIZmiZPyLGQBGRYIC
Y2ExGTAXBgoJkiaJk/IsZAEZFGlzYW5kZWxtyW4xHDAaBgNVBAEMVE1v3PlyzW5n
IEhpZ2h3YxkgQ0ewHhcNMTcwMzI2MTYxOThwNHMNMtcwMzI2MTYxOThwNHMNMtc
EAYKzIZmiZPyLGQBGRYICY2ExGTAXBgoJkiaJk/IsZAEZFGlzYW5kZWxtyW4xKJAU
BgNVBAEHMDVUc3PlyzW5nIIEI1UOwA6gQcghkjOPQIIBBeUQAIgNiAATZAH3R
b2FvJIOnts+vxuWW35ofyNbcHzzjAz0I2kWFZFE1ByurNmcNMFGiGrGnRXIGqWCf
w5ICgJ8CuM3v5ty9bf7KU0kejzTv+5PV++elkP9H9Q83vqTaws2WwWTxJEDAO
MAwGA1UdEBodEwEB/wQCMMAAwCgYIKwYBBQUHAwEGCCsGAQUFBzAChGZDQGCC
-----END CERTIFICATE-----
```

D.1.2. Manufacturer key pair for IDevID signatures

This private key signs IDevID certificates:

```
-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQDAjBNNMRIwEAYKzIZmiZPyLGQBGRYIC
Y2ExGTAXBgoJkiaJk/IsZAEZFGlzYW5kZWxtyW4xHDAaBgNVBAEMVE1v3PlyzW5n
IEhpZ2h3YxkgQ0ewHhcNMTcwMzI2MTYxOThwNHMNMtcwMzI2MTYxOThwNHMNMtc
EAYKzIZmiZPyLGQBGRYICY2ExGTAXBgoJkiaJk/IsZAEZFGlzYW5kZWxtyW4xKJAU
BgNVBAEHMDVUc3PlyzW5nIIEI1UOwA6gQcghkjOPQIIBBeUQAIgNiAATZAH3R
b2FvJIOnts+vxuWW35ofyNbcHzzjAz0I2kWFZFE1ByurNmcNMFGiGrGnRXIGqWCf
w5ICgJ8CuM3v5ty9bf7KU0kejzTv+5PV++elkP9H9Q83vqTaws2WwWTxJEDAO
MAwGA1UdEBodEwEB/wQCMMAAwCgYIKwYBBQUHAwEGCCsGAQUFBzAChGZDQGCC
-----END CERTIFICATE-----
```

Pritikin, et al. Expires December 19, 2019
This public key validates IDevID certificates:

-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBgkqhjIwQDQDAjBNNMIwEAYKCZIMizPylQGQBGRYCY
Y2ExGTAXBgoJkiaJK/IzZAEZ9Fg1zYW5kZWxhYWxhHDAAbGNgVBAMMFE1Vuc3IydW5n
IEhpZ2h3YXkgQDEwHhcNMTcwMzI2MTYxOTQwHWhcNMTkwMzI2MTYxOTQwHBJHMIw
EAYKCZIMizPylQGQBGRYCYZExGTAXBgoJkiaJK/IzZAEZ9Fg1zYW5kZWxhYWxh+JAY
BgNGVBAMMDVc3IydW5nIE1BIUOEwdjaQBgcghkjoOPQIBB0urgQQAiNiAATZAH3R
b2FjJ0nts+vXuW35ofyNhBhjjaZoI2kW2FE1byrKimNCNM5FilarGnRXXGQxwCf
w51IQgCjuM3v5ty9b7fKUN0kejzTvV+5PV++elkP9HQ83vQTaws2WWWTxI=
-----END CERTIFICATE-----

D.1.3. Registrar key pair

The registrar key (or chain) is the representative of the domain owner. This key signs registrar voucher-requests:

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIF+obiToYYyMifPs3vrjWj0yFsCJwIFhpoKM7/TULmXoAOGCCqGSM49
AwEhoQUQDQAENWQozNMMO5PntFeBEODJ/LwfeMGyDFIdvF6Uz4DIFM1ujMBec/g
6W/P6bOnmyTGdFOh/8hwKUerL5bpeK8sg==
-----END EC PRIVATE KEY-----

The public key is indicated in a pledge voucher-request to show proximity.

-----BEGIN CERTIFICATE-----
MIIBrjCCATQgAwIBAgIAlBZKAggbqghkjoOPQOAzBOMRlWeEAYKCZIMizPylQGQBGRYCY
Y2ExGTAXBgoJkiaJK/IzZAEZ9Fg1zYW5kZWxhYWxhHDAAbGNgVBAMMFE1Vuc3IydW5n
IEhpZ2h3YXkgQDEwHhcNMTcwMzI2MTYxOTQwHWhcNMTkwMzI2MTYxOTQwHBJHMIw
EAYKCZIMizPylQGQBGRYCYZExGTAXBgoJkiaJK/IzZAEZ9Fg1zYW5kZWxhYWxh+JAY
BgNGVBAMMDVc3IydW5nIE1BIUOEwdjaQBgcghkjoOPQIBB0urgQQAiNiAATZAH3R
b2FjJ0nts+vXuW35ofyNhBhjjaZoI2kW2FE1byrKimNCNM5FilarGnRXXGQxwCf
w51IQgCjuM3v5ty9b7fKUN0kejzTvV+5PV++elkP9HQ83vQTaws2WWWTxI=
-----END CERTIFICATE-----

The public key is indicated in a pledge voucher-request to show proximity.

-----BEGIN CERTIFICATE-----
MIIBrjCCATQgAwIBAgIAlBZKAggbqghkjoOPQOAzBOMRlWeEAYKCZIMizPylQGQBGRYCY
Y2ExGTAXBgoJkiaJK/IzZAEZ9Fg1zYW5kZWxhYWxhHDAAbGNgVBAMMFE1Vuc3IydW5n
IEhpZ2h3YXkgQDEwHhcNMTcwMzI2MTYxOTQwHWhcNMTkwMzI2MTYxOTQwHBJHMIw
EAYKCZIMizPylQGQBGRYCYZExGTAXBgoJkiaJK/IzZAEZ9Fg1zYW5kZWxhYWxh+JAY
BgNGVBAMMDVc3IydW5nIE1BIUOEwdjaQBgcghkjoOPQIBB0urgQQAiNiAATZAH3R
b2FjJ0nts+vXuW35ofyNhBhjjaZoI2kW2FE1byrKimNCNM5FilarGnRXXGQxwCf
w51IQgCjuM3v5ty9b7fKUN0kejzTvV+5PV++elkP9HQ83vQTaws2WWWTxI=
-----END CERTIFICATE-----

The registrar public certificate as decoded by openssl’s x509 utility. Note that the registrar certificate is marked with the cmcRA extension.

Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 3 (0x3)
  Signature Algorithm: ecdsa-with-SHA384
  Issuer: DC = ca, DC = sandelman, CN = Unstrung Fountain CA

Validity
  Not Before: Sep 5 01:12:45 2017 GMT
  Not After : Sep 5 01:12:45 2019 GMT
  Subject: DC = ca, DC = sandelman, CN = localhost
  Subject Public Key Info:
  Public Key Algorithm: id-ecPublicKey
    PublicKey: (256 bit)
    pub:
      e9:9d:e2:bc:b2
    ASN1 OID: prime256v1
    NIST CURVE: P-256
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  Signature Algorithm: ecdsa-with-SHA384
D.1.4. Pledge key pair

The pledge has an IDevID key pair built in at manufacturing time:

```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBgR6SV+uEvWf15zCQWXzWjYbMhXPYqNdHJ3KPh11mm4oAoGCCqGSM49
AwEHoUQDgqAEwi/jgqPpRj0JgWh2RgeZlKutbXVjmnHb+1AYaEF/YQjEzg5FZV8
KjiR/bkEl+18M4onIC7RKaXXKkuag9S6Tw==
-----END EC PRIVATE KEY-----
```

The public key is used by the registrar to find the MASA. The MASA URL is in an extension described in Section 2.3.

```
-----BEGIN CERTIFICATE-----
MIICBDCCAYugAwIBAgIECe20qTAKBgqghkjiOPQDAjBNMRIwEAYKCZIimi2PyLQGB
GRVCY2ExGTAXBgojKxKajKj/IsZAEXFglzYW5kZWxtYW4xHDAaBgNVAMME1Vuc3Ry
dW5n1EhpzZ3YkgqQwIEhBcNMTkwNDI0MDIxNjU4WgqPMj50TEyMzEwMDAwMDAa
MBxw/jAYBgNVBAUETmFEBWFiYm9Hc3BwY2h5by5wLmJpZ2h3cy5zc3QgL0NB
ziQOcDqgAEWI/jgqPpRj0JgWh2RgeZlKutbXVjmnHb+1AYaEF/YQjEzg5FZV8
KjiR/bkEl+18M4onIC7RKaXXKkuag9S6Tw==
-----END CERTIFICATE-----
```

The pledge public certificate as decoded by openssl’s x509 utility so that the extensions can be seen. There is a second Custom Extension is included to provided to contain the EUI48/EUI64 that the pledge will configure as it’s layer-2 address (this is non-normative).
Certificate:

Data:
Version: 3 (0x2)
Serial Number: 166573225 (0x9edb4a9)
Signature Algorithm: ecdsa-with-SHA256
Issuer: DC = ca, DC = sandelman, CN = Unstrung Highway CA

Validity
Not Before: Apr 24 02:16:58 2019 GMT
Not After : Dec 31 00:00:00 2999 GMT
Subject: serialNumber = 00-d0-e5-02-00-2d

Subject Public Key Info:
Public Key Algorithm: id-ecPublicKey
Public-Key: (256 bit)
pub:
9a:83:d4:ba:4f
ASN1 OID: prime256v1
NIST CURVE: P-256

X509v3 extensions:
X509v3 Subject Key Identifier:
X509v3 Basic Constraints:
CA:FALSE
X509v3 Subject Alternative Name:
othername:<unsupported>
1.3.6.1.4.1.46930.2:
..masa.honeydukes.sandelman.ca

Signature Algorithm: ecdsa-with-SHA256
16:ce:3f:54:55:a0:54:e5:0d:0b:8e:ff:79:8b:cd:be:64:53:

D.2. Example process

RFC-EDITOR: these examples will need to be replaced with CMS versions once IANA has assigned the eContentType in [RFC8366].

D.2.1. Pledge to Registrar

As described in Section 5.2, the pledge will sign a pledge voucher-request containing the registrar’s public key in the proximity-
The ASN.1 decoding of the artifact:

0:d=0  hl=4 l=1717 cons: SEQUENCE
 4:d=1  hl=2 l=   9 prim: OBJECT :pkcs7-signedData

file: examples/vr_00-D0-E5-02-00-2D.pkcs

The ASN1 decoding of the artifact:

0:d=0  hl=4 l=1717 cons: SEQUENCE
 4:d=1  hl=2 l=  9 prim: OBJECT :pkcs7-signedData
15:d=1  hl=4  l=1702 cons:  cont [ 0 ]
19:d=2  hl=4  l=1698 cons:  SEQUENCE
23:d=3  hl=2  l=  1 prim:  INTEGER   :01
26:d=3  hl=2  l= 13 cons:  SET
28:d=4  hl=2  l= 11 cons:  SEQUENCE
30:d=5  hl=2  l=  9 prim:  OBJECT    :sha256
41:d=3  hl=4  l= 849 cons:  SEQUENCE
45:d=4  hl=2  l=  9 prim:  OBJECT    :pkcs7-data
56:d=4  hl=4  l= 834 cons:  cont [ 0 ]
60:d=5  hl=4  l= 830 prim:  OCTET STRING: "ietf-voucher-request:ver...
894:d=3  hl=4  l= 520 cons:  cont [ 0 ]
898:d=4  hl=4  l= 516 cons:  SEQUENCE
902:d=5  hl=4  l= 395 cons:  SEQUENCE
906:d=6  hl=2  l=   3 cons:  cont [ 0 ]
908:d-7  hl=2  l=   1 prim:  INTEGER   :02
911:d=6  hl=2  l=  4 prim:  INTEGER   :09EDB4A9
917:d=6  hl=2  l= 10 cons:  SEQUENCE
919:d=7  hl=2  l=  8 prim:  OBJECT    :ecdsa-with-SHA256
929:d=6  hl=2  l= 77 cons:  SEQUENCE
931:d=7  hl=2  l= 18 cons:  SET
933:d=8  hl=2  l= 16 cons:  SEQUENCE
935:d=9  hl=2  l= 10 prim:  OBJECT    :domainComponent
947:d=9  hl=2  l=   2 prim:  IA5STRING :ca
951:d=7  hl=2  l=  25 cons:  SET
953:d=8  hl=2  l=  23 cons:  SEQUENCE
955:d=9  hl=2  l=  10 prim:  OBJECT    :domainComponent
967:d=9  hl=2  l=  9 prim:  IA5STRING :sandelman
978:d=7  hl=2  l=  28 cons:  SET
980:d=8  hl=2  l= 26 cons:  SEQUENCE
982:d=9  hl=2  l=  3 prim:  OBJECT    :commonName
987:d=9  hl=2  l= 19 prim:  UTF8STRING :Unstrung Highway CA
1008:d=6  hl=2  l= 32 cons:  SEQUENCE
1010:d=7  hl=2  l= 13 prim:  UTCTIME   :190424021658Z
1025:d=7  hl=2  l= 15 prim:  GENERALIZEDTIME :29991231000000Z
1042:d=6  hl=2  l= 28 cons:  SEQUENCE
1044:d=7  hl=2  l= 26 cons:  SET
1046:d=8  hl=2  l= 24 cons:  SEQUENCE
1048:d=9  hl=2  l=   3 prim:  OBJECT    :serialNumber
1053:d=9  hl=2  l=  17 prim:  UTF8STRING :00-d0-e5-02-00-2d
1072:d=6  hl=2  l= 89 cons:  SEQUENCE
1074:d=7  hl=2  l= 19 cons:  SEQUENCE
1076:d=8  hl=2  l=   7 prim:  OBJECT    :id-ecPublicKey
1085:d=8  hl=2  l=   8 prim:  OBJECT    :prime256v1
1095:d=7  hl=2  l=  66 prim:  BIT STRING
1163:d=6  hl=3  l= 135 cons:  cont [ 3 ]
1166:d=7  hl=3  l= 132 cons:  SEQUENCE
1169:d=8  hl=2  l= 29 cons:  SEQUENCE
1171:d=9  hl=2  l=  3 prim:  OBJECT    :X509v3 Subject Key Ident
1176:d=9  hl=2 l=  22 prim: OCTET STRING  [HEX DUMP]:04148FC298754A
1200:d=8  hl=2 l=   9 cons:  SEQUENCE
1202:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
1207:d=9  hl=2 l=   2 prim: OCTET STRING  [HEX DUMP]:3000
1211:d=8  hl=2 l=  43 cons:  SEQUENCE
1213:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Alternative
1218:d=9  hl=2 l=  36 prim: OCTET STRING  [HEX DUMP]:3022A02006092B
1256:d=8  hl=2 l=  43 cons:  SEQUENCE
1258:d=9  hl=2 l=   9 prim: OBJECT            :1.3.6.1.4.1.46930.2
1269:d=9  hl=2 l=  30 prim: OCTET STRING  [HEX DUMP]:0C1C6D6173612E
1301:d=5  hl=2 l=  10 cons:  SEQUENCE
1303:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
1313:d=5  hl=2 l=  103 prim:  BIT STRING
1418:d=3  hl=4 l=  299 cons:  SET
1422:d=4  hl=4 l=  295 cons:  SEQUENCE
1426:d=5  hl=2 l=   1 prim: INTEGER           :01
1429:d=5  hl=2 l=  85 cons:  SEQUENCE
1431:d=6  hl=2 l=   77 cons:  SEQUENCE
1433:d=7  hl=2 l=  18 cons:  SET
1435:d=8  hl=2 l=  16 cons:  SEQUENCE
1437:d=9  hl=2 l=  10 prim:  OBJECT            :domainComponent
1449:d=9  hl=2 l=   2 prim: IA5STRING         :ca
1453:d=7  hl=2 l=   25 cons:  SET
1455:d=8  hl=2 l=  23 cons:  SEQUENCE
1457:d=9  hl=2 l=  10 prim:  OBJECT            :domainComponent
1469:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
1480:d=7  hl=2 l=  28 cons:  SET
1482:d=8  hl=2 l=  26 cons:  SEQUENCE
1484:d=9  hl=2 l=   3 prim:  OBJECT            :commonName
1489:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
1510:d=6  hl=2 l=   4 prim: INTEGER           :09EDB4A9
1516:d=5  hl=2 l=  11 cons:  SEQUENCE
1518:d=6  hl=2 l=  17 cons:  SEQUENCE
1529:d=5  hl=2 l=  105 cons:  cont [ 0 ]
1531:d=6  hl=2 l=  24 cons:  SEQUENCE
1533:d=7  hl=2 l=   9 prim:  OBJECT            :contentType
1544:d=7  hl=2 l=  11 cons:  SET
1546:d=8  hl=2 l=   9 prim:  OBJECT            :pkcs7-data
1557:d=6  hl=2 l=  28 cons:  SEQUENCE
1559:d=7  hl=2 l=  19 cons:  OBJECT            :signingTime
1570:d=7  hl=2 l=  15 cons:  SET
1572:d=8  hl=2 l=  13 prim: UTCTIME           :190515212555Z
1587:d=6  hl=2 l=  47 cons:  SEQUENCE
1589:d=7  hl=2 l=   9 prim:  OBJECT            :messageDigest
1600:d=7  hl=2 l=  34 cons:  SET
1602:d=8  hl=2 l=  32 prim: OCTET STRING  [HEX DUMP]:1037694FEDAAB0
1636:d=5  hl=2 l=  10 cons:  SEQUENCE
1638:d=6  hl=2 l=   8 prim:  OBJECT            :ecdsa-with-SHA256

The JSON contained in the voucher request:

```
{
  "ietf-voucher-request:voucher": {
    "assertion": "proximity",
    "created-on": "2019-05-15T17:25:55.644-04:00",
    "serial-number": "00-d0-e5-02-00-2d",
    "nonce": "VOUFT-WwrEv0uAQEHoV7Q",
    "proximity-registrar-cert": "MIIB0TCCAVagAwIBAgIBAjAKBggqhkjOPQQDAzBxMlo=",
    "proximity-registrar-cert": "MIIB0TCCAVagAwIBAgIBAjAKBggqhkjOPQQDAzBxMlo=
```

As described in Section 5.5 the registrar will sign a registrar voucher-request, and will include pledge’s voucher request in the prior-signed-voucher-request.

-----BEGIN CMS-----
MIIPkwYJKoZIhvcNAQcCoIIPhDCCD4ACAQExDTALBglghkgBZQMEAgEwggnUBgkq
hkiG9w0BBwGggbhkiG9w0BBwGggbhkiG9w0BBwGggbhkiG9w0BBwGggbhkiG9w0BBwG
gbhkiG9w0BBwGggbhkiG9w0BBwGggbhkiG9w0BBwGggbhkiG9w0BBwGggbhkiG9w0BBwG

```
The ASN1 decoding of the artifact:

```
0:d=0  hl=4 l=3987 cons: SEQUENCE
  4:d=1  hl=2 l=  9 prim: OBJECT :pkcs7-signedData
15:d=1  hl=2 l=3972 cons: cont [ 0 ]
19:d=2  hl=2 l=3968 cons: SEQUENCE
23:d=3  hl=2 l=  1 prim: INTEGER :01
26:d=3  hl=2 l= 13 cons: SET
28:d=4  hl=2 l= 11 cons: SEQUENCE
30:d=5  hl=2 l=  9 prim: OBJECT :sha256
41:d=3  hl=4 l=2516 cons: SEQUENCE
45:d=4  hl=4 l=2501 cons: cont [ 0 ]
56:d=4  hl=2 l=2501 cons: cont [ 0 ]
60:d=5  hl=4 l=2497 prim: OCTET STRING :{"ietf-voucher-request:v
2561:d=3  hl=4 l=1090 cons: cont [ 0 ]
2565:d=4  hl=4 l= 465 cons: SEQUENCE
2569:d=5  hl=4 l= 342 cons: SEQUENCE
2573:d=6  hl=2 l=  3 cons: cont [ 0 ]
2575:d=7  hl=2 l=  1 prim: INTEGER :02
2578:d=6  hl=2 l=  1 prim: INTEGER :02
2581:d=6  hl=2 l= 10 cons: SEQUENCE
2583:d=7  hl=2 l=  8 prim: OBJECT :ecdsa-with-SHA384
2593:d=6  hl=2 l=113 cons: SEQUENCE
2595:d=7  hl=2 l= 18 cons: SET
2597:d=8  hl=2 l= 16 cons: SEQUENCE
2599:d=9  hl=2 l= 10 prim: OBJECT :domainComponent
2611:d=9  hl=2 l=  2 prim: IA5STRING :ca
2615:d=7  hl=2 l=  5 cons: SET
2617:d=8  hl=2 l= 23 cons: SEQUENCE
2619:d=9  hl=2 l= 10 prim: OBJECT :domainComponent
2631:d=9  hl=2 l=  9 prim: IA5STRING :sandelman
2642:d=7  hl=2 l= 64 cons: SET
2644:d=8  hl=2 l= 62 cons: SEQUENCE
2646:d=9  hl=2 l=  3 prim: OBJECT :commonName
2651:d=9  hl=2 l= 55 prim: UTF8STRING :#<SystemVariable:0x00000
2708:d=6  hl=2 l= 30 cons: SEQUENCE
2710:d=7  hl=2 l= 13 prim: UTCTIME :171107234528Z
2725:d=7  hl=2 l= 13 prim: UTCTIME :191107234528Z
2740:d=6  hl=2 l=  67 cons: SEQUENCE
2742:d=7  hl=2 l=  18 cons: SET
2744:d=8  hl=2 l= 16 cons: SEQUENCE
2746:d=9  hl=2 l= 10 prim: OBJECT :domainComponent
2758:d=9  hl=2 l=  2 prim: IA5STRING :ca
```
2762:d=7  hl=2 l=  25 cons: SET
2764:d=8  hl=2 l=  23 cons: SEQUENCE
2766:d=9  hl=2 l=  10 prim: OBJECT :domainComponent
2778:d=9  hl=2 l=   9 prim: IA5STRING :sandelman
2789:d=7  hl=2 l=  18 cons: SET
2791:d=8  hl=2 l=  16 cons: SEQUENCE
2793:d=9  hl=2 l=   3 prim: OBJECT :commonName
2798:d=9  hl=2 l=  10 prim: UTF8STRING :localhost
2809:d=6  hl=2 l=  89 cons: SEQUENCE
2811:d=7  hl=2 l=  19 cons: SEQUENCE
2813:d=8  hl=2 l=    7 prim: OBJECT :id-ecPublicKey
2822:d=8  hl=2 l=    8 prim: OBJECT :prime256v1
2832:d=7  hl=2 l=   66 prim: BIT STRING
2900:d=6  hl=2 l=  13 cons: cont [ 3 ]
2902:d=7  hl=2 l=   11 cons: SEQUENCE
2904:d=8  hl=2 l=    9 cons: SEQUENCE
2906:d=9  hl=2 l=    3 prim: OBJECT :X509v3 Basic Constraints
2911:d=9  hl=2 l=    2 prim: OCTET STRING [HEX DUMP]:3000
2915:d=5  hl=2 l=    10 cons: SEQUENCE
2917:d=6  hl=2 l=    8 prim: OBJECT :ecdsa-with-SHA384
2927:d=5  hl=2 l=  105 prim: BIT STRING
3034:d=4  hl=1 l=  617 cons: SEQUENCE
3038:d=5  hl=1 l=  495 cons: SEQUENCE
3042:d=6  hl=2 l=    3 cons: cont [ 0 ]
3044:d=7  hl=2 l=    1 prim: INTEGER :02
3047:d=6  hl=2 l=    1 prim: INTEGER :03
3050:d=6  hl=2 l=    10 cons: SEQUENCE
3052:d=7  hl=2 l=    8 prim: OBJECT :ecdsa-with-SHA256
3062:d=6  hl=2 l=  109 cons: SEQUENCE
3064:d=7  hl=2 l=    18 cons: SET
3066:d=8  hl=2 l=    16 cons: SEQUENCE
3068:d=9  hl=2 l=    10 prim: OBJECT :domainComponent
3080:d=9  hl=2 l=    2 prim: IA5STRING :ca
3084:d=7  hl=2 l=    25 cons: SET
3086:d=8  hl=2 l=    23 cons: SEQUENCE
3088:d=9  hl=2 l=    10 prim: OBJECT :domainComponent
3100:d=9  hl=2 l=    9 prim: IA5STRING :sandelman
3111:d=7  hl=2 l=    60 cons: SET
3113:d=8  hl=2 l=    58 cons: SEQUENCE
3115:d=9  hl=2 l=    3 prim: OBJECT :commonName
3120:d=9  hl=2 l=   51 prim: UTF8STRING :fountain-test.example.co
3173:d=6  hl=2 l=    30 cons: SEQUENCE
3175:d=7  hl=2 l=    13 prim: UTCTIME :190113225444Z
3190:d=7  hl=2 l=    13 prim: UTCTIME :210112225444Z
3205:d=6  hl=2 l=   109 cons: SEQUENCE
3207:d=7  hl=2 l=    18 cons: SET
3209:d=8  hl=2 l=    16 cons: SEQUENCE
3211:d=9  hl=2 l=    10 prim: OBJECT :domainComponent
D.2.3. MASA to Registrar

The MASA will return a voucher to the registrar, to be relayed to the pledge.
Beginning CMS

-----BEGIN CMS-----

MIIGsgYJKoZIhvcNAQcCoIIGozCCBp8CAQExDTALBglghkgBZQMEAgEwggNAhBqgk
hkiG9w0BBgwGggbMbxIDIXsiaWV0Z12b3VjaGVyOnZvdWNOZXIiOnsiYXNzZXJ0
aW9uIjoiBn9zZVIIwiYi1JYK1ZCviBhI61i1jWmtkMTUtMTZUMTQ1MTI1NDBNi
NjkK3KAw0JaWiic2VyaWFsLWli5bWJlci1i6jJi1mAQjLVQWLU1TAYT0aLJTkji1w
b9yU0iOi1JYWtmTt2pYjw5N0V2NNNIN6Uz1iWiicGlubmVlLW1wWFpbpij7
ZJX0juijTU1j1jQBUQNBVnFQxjDQjK 예산FJnkMndcWhrk4RUVFQXCPe1S
SxfDQVQL1QjPBiWla1UH1M1FCR1JQ1kyRXhVHEFYQmdvSmpUTyprL0izWkFFWkZm
bHPVZwFLd4dFI1XNhHRERQmqcOKJBT10eU04VTNsemRHvRwBzU5YDvgAaWJH
VT2NSGd3TURBd01EQx0QIrK1tVVGa1ENGDWVzS6EhKNWJyt2DSbTkXmISaGFX
NGDRNV3S9NjhiTk1yj3nHVEZtTwpmN5JUXTRAxGNOTVRrE1UQTNnakwT1RJNFdq
QKRNuk13RUFZS0Nas1pP15dEdRQkdSWU9MV41RBEWEJnb0p0praPFWaY9jc1ipB
RypGz2x6W6c1aipXeHXRvz4x4WRpBUUJnT1zCQ1U1Qd4dkyInNhRs16ZERCWk1C
TudCeFUX00000OUfnRdDQ3FU0000UF3UheBMElBqkPbWFVISTB1c9s2MaVz1j2
QoJiKzXjbM9FTVnuYzdSbytYwkN0akFUMENEMWZKz2ks2L2heX1eBuhXeVlptKzi
UkHIOW25YXuMa3pNdRWWmP6pUakxakRQUXuN0Q0THP7FV7ZVE3UUNQFQ3Q2d5SUtv
Wk16ajFpqXNdRGRFQx0da2Z04Q0xRTU50cmY4DHY1MGxSTQ1RFFYSEVSpk0VzKR
VjJn0SFVE2ZERTazJWNStBi1nY1q1tR1Oa00gm02bxFT2hFdxUnxSNhBSJrV22OdytC
amJaBtp5SW1tRWNUN0h1oEnSWGFNSFvRjduMz130tjOjKT1T25tK1xB3PBRUXs
Nm3xM0NacVE9PSJ9faCCAFuwgHxMIIBeKADAeCAgQjZjKTMa0xGCG0SGM49BAMC
ME0ExjQBOgjKiajK/IsZAE2FGjYJYTEZMBCCgCmJSomT8iXkARKWCXhbbRbbOh1h
bjeBMcBGAuAwTwTV5dhi1JmcmqG1naHdheSBQ7AeFw0oxOTA0jMyMzIXMDda
Fw0oxOTaMjQotOTIXmMddMgyDXAnBqNVBAyTBKnhbpmFKYESMBAGA1UECgwU2FU
ZGVSbWFuMRMWXEQYDVQQLApob251eNRa2VzMsowKAYDVQQDCAFXYXNhbmVbV5
ZHVzXMu2fUZGswbWFwLnMnRiEBU0OewdJABqcchkJ0POIQ1BgUrqOQAIIx4A1Q
/2UdpVZvMgAD0nBq17LcP3Lse3aVAdgEYqSAbikNko0TO30jP10qNBxtGFRFBXx
ghzKhTH5bR85Wt/MejSAbQhB4S2yyjwMWRUd71J0ummrtRwtf7+OgAEfewCj
EDOMAVGALuidwEB/0XCMATAcYq1ZiZj0EAdIDZwAw0zOM1NOMNYSEoz4ylW4
iR1tLBuirmJvdMfVn1yZsHd1snj0pa3xQqKZ55LLARoSRxobTqxMasnx6ySHAp2
I WCspDZ2G1OSDpMn7nuRJSdkgWqexvIL4+9nml5sFMbDvz1DMhAxdgsFM1MIBSA
ATBVMeOxjQBojKiajK/IsZAE2FGjYJYTEZMBCCgCmJSomT8iXkARKWCXhbbRbbRl
bGl1hbjeBMcBGAuAwTwTV5dhi1JmcmqG1naHdheSBQ7QIE18yJExALBlqhgkB
ZQMEAgGaaTAYbqkgkhig90wBCqMcxyYkoZ1hcvNacQMcCwBGScGScG3DqJEBTEP
Fw0oxOTA1TMWMyjuxiNdJaMC8CSqGiSB3DEQjEBEiBCCYHR412iQjjeek8leRLVX
/ev5Yg1b4MO4Q210z84CDARKBggjkhjQOPQDAgoMGYCMQCYYO11B1L4Ed4nAnOl
a4S81XWA295XGp77bG/t4F4ETTVT353nMeeYBeNfhC6/kOCMQOq1LcmwQJQzDEl
asj1Slnj/FnZjiGOMz59MX0mGNGfW9v2VBb9mVyhO8McqlVig=

------END CMS------

file: examples/voucher_00-D0-E5-02-00-2D.pkcs

The ASN1 decoding of the artifact:

0:d=0 hl=4 l=1714 cons: SEQUENCE
4:d=1 hl=2 l= 9 prim: OBJECT :pkcs7-signedData
15:d=0 hl=3 l= 0 cons: cont [0]
19:d=2 hl=2 l=150 cons: INTEGER :10
23:d=3 hl=2 l= 0 cons: finish [0]

26:d=3  hl=2 l=  13 cons:  SET
28:d=4  hl=2 l=  11 cons:  SEQUENCE
30:d=5  hl=2 l=   9 prim:  OBJECT :sha256
41:d=3  hl=4 l=  82 cons:  SEQUENCE
45:d=4  hl=2 l=   9 prim:  OBJECT :pkcs7-data
56:d=4  hl=4 l=  817 cons: cont [ 0 ]
60:d=5  hl=4 l=  813 prim: OCTET STRING :{"ietf-voucher:voucher":
877:d=3  hl=4 l=  501 cons: cont [ 0 ]
881:d=4  hl=4 l=  497 cons:  SEQUENCE
885:d=5  hl=4 l=  376 cons:  SEQUENCE
889:d=6  hl=2 l=   3 cons:  cont [ 0 ]
891:d=7  hl=2 l=   1 prim:  INTEGER :02
894:d=6  hl=2 l=   4 prim:  INTEGER :23CC8913
900:d=6  hl=2 l=  10 cons:  SEQUENCE
902:d=7  hl=2 l=   8 prim:  OBJECT :ecdsa-with-SHA256
912:d=6  hl=2 l=  77 cons:  SEQUENCE
914:d=7  hl=2 l=  18 cons:  SET
916:d=8  hl=2 l=  16 cons:  SEQUENCE
918:d=9  hl=2 l=  10 prim:  OBJECT :domainComponent
930:d=9  hl=2 l=   2 prim:  IA5STRING :ca
934:d=7  hl=2 l=  25 cons:  SET
936:d=8  hl=2 l=  23 cons:  SEQUENCE
938:d=9  hl=2 l=  10 prim:  OBJECT :domainComponent
950:d=9  hl=2 l=   9 prim:  IA5STRING :sandelman
961:d=7  hl=2 l=  28 cons:  SET
963:d=8  hl=2 l=  26 cons:  SEQUENCE
965:d=9  hl=2 l=   3 prim:  OBJECT :commonName
970:d=9  hl=2 l=  19 prim:  UTF8STRING :Unstrung Highway CA
991:d=6  hl=2 l=  30 cons:  SEQUENCE
993:d=7  hl=2 l=  13 prim:  UTCTIME :190423232107Z
1008:d=7  hl=2 l=  13 prim:  UTCTIME :190524092107Z
1023:d=6  hl=2 l=  102 cons:  SEQUENCE
1025:d=7  hl=2 l=  15 cons:  SET
1027:d=8  hl=2 l=  13 cons:  SEQUENCE
1029:d=9  hl=2 l=   3 prim:  OBJECT :countryName
1034:d=9  hl=2 l=   6 prim:  PRINTABLESTRING :Canada
1042:d=7  hl=2 l=  18 cons:  SET
1044:d=8  hl=2 l=  16 cons:  SEQUENCE
1046:d=9  hl=2 l=   3 prim:  OBJECT :organizationName
1051:d=9  hl=2 l=   9 prim:  UTF8STRING :Sandelman
1062:d=7  hl=2 l=  19 cons:  SET
1064:d=8  hl=2 l=  17 cons:  SEQUENCE
1066:d=9  hl=2 l=   3 prim:  OBJECT :organizationalUnitName
1071:d=9  hl=2 l=  10 prim:  UTF8STRING :honeydukes
1083:d=7  hl=2 l=  42 cons:  SET
1085:d=8  hl=2 l=  40 cons:  SEQUENCE
1087:d=9  hl=2 l=   3 prim:  OBJECT :commonName
1092:d=9  hl=2 l=  33 prim:  UTF8STRING :masa.honeydukes.sandelma
Internet-Draft                    BRSKI                        June 2019

1127:d=6  hl=2 l= 118 cons: SEQUENCE
1129:d=7  hl=2 l=  16 cons: SEQUENCE
1131:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
1140:d=8  hl=2 l=   5 prim: OBJECT            :secp384r1
1147:d=7  hl=2 l=  98 prim: BIT STRING
1247:d=6  hl=2 l=  16 cons: cont [ 3 ]
1249:d=7  hl=2 l=  14 cons: SEQUENCE
1251:d=8  hl=2 l=  12 cons: SEQUENCE
1253:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
1258:d=9  hl=2 l=   1 prim: BOOLEAN           :255
1261:d=9  hl=2 l=   2 prim: OCTET STRING [HEX DUMP]:3000
1265:d=5  hl=2 l=  10 cons: SEQUENCE
1267:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
1277:d=5  hl=2 l= 103 cons: SEQUENCE
1382:d=3  hl=4 l= 323 cons: SEQUENCE
1386:d=4  hl=2 l= 328 cons: SEQUENCE
1390:d=5  hl=2 l=   1 prim: INTEGER           :01
1393:d=5  hl=2 l=  85 cons: SEQUENCE
1395:d=6  hl=2 l=  77 cons: SEQUENCE
1397:d=7  hl=2 l=  18 cons: SET
1399:d=8  hl=2 l=  16 cons: SEQUENCE
1401:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
1413:d=9  hl=2 l=   2 prim: IA5STRING         :ca
1417:d=7  hl=2 l=  25 cons: SET
1421:d=9  hl=2 l=  23 cons: SEQUENCE
1429:d=9  hl=2 l=  19 prim: IA5STRING         :sandelman
1444:d=7  hl=2 l=  28 cons: SET
1446:d=8  hl=2 l=  26 cons: SEQUENCE
1448:d=9  hl=2 l=   3 prim: OBJECT            :commonName
1453:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
1474:d=6  hl=2 l=   4 prim: INTEGER           :23CC8913
1480:d=5  hl=2 l=  11 cons: SEQUENCE
1482:d=6  hl=2 l=   9 prim: OBJECT            :sha256
1493:d=5  hl=2 l= 105 cons: cont [ 0 ]
1495:d=6  hl=2 l=  24 cons: SEQUENCE
1497:d=7  hl=2 l=   9 prim: OBJECT            :contentType
1508:d=7  hl=2 l=  11 cons: SET
1510:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
1521:d=6  hl=2 l=  28 cons: SEQUENCE
1523:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
1534:d=7  hl=2 l=  15 cons: SET
1536:d=8  hl=2 l=  13 prim: UTCTIME           :190516025142Z
1551:d=6  hl=2 l=  47 cons: SEQUENCE
1553:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
1564:d=7  hl=2 l=  34 cons: SET
1566:d=8  hl=2 l=  32 prim: OCTET STRING [HEX DUMP]:98461E22DB5423
1600:d=5  hl=2 l=  10 cons: SEQUENCE

1602:d=6  hl=2 l=   8 prim: OBJECT :ecdsa-with-SHA256
1612:d=5  hl=2 l= 104 prim: OCTET STRING [HEX DUMP]:30660231009860

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

Email: Michael.H.Behringer@gmail.com

Steinthor Bjarnason
Arbor Networks

Email: sbjarnason@arbor.net

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net
Abstract

This document defines a strategy to securely assign a pledge to an owner, using an artifact signed, directly or indirectly, by the pledge's manufacturer. This artifact is known as a "voucher".

This document builds upon the work in [RFC8366], encoding the resulting artifact in CBOR. Use with two signature technologies are described.

Additionally, this document explains how constrained vouchers may be transported in the [I-D.ietf-ace-coap-est] protocol.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 26, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................... 4
3. Requirements Language ......................................... 4
4. Survey of Voucher Types ....................................... 4
5. Discovery and URI .............................................. 5
6. Artifacts ....................................................... 6
   6.1. Voucher Request artifact ................................ 7
       6.1.1. Tree Diagram ......................................... 7
       6.1.2. SID values .......................................... 7
       6.1.3. YANG Module ......................................... 8
       6.1.4. Example voucher request artifact .................... 12
   6.2. Voucher artifact ........................................... 12
       6.2.1. Tree Diagram ......................................... 13
       6.2.2. SID values .......................................... 13
       6.2.3. YANG Module ......................................... 14
       6.2.4. Example voucher artifacts ......................... 16
   6.3. CMS format voucher and voucher-request artifacts ........ 16
       6.3.1. COSE signing ......................................... 17
7. Design Considerations .......................................... 18
8. Security Considerations ........................................ 18
   8.1. Clock Sensitivity ......................................... 18
   8.2. Protect Voucher PKI in HSM .............................. 18
   8.3. Test Domain Certificate Validity when Signing .......... 18
9. IANA Considerations ............................................ 18
   9.1. Resource Type Registry .................................. 18
   9.2. The IETF XML Registry ................................... 18
   9.3. The YANG Module Names Registry .......................... 19
   9.4. The SMI Security for S/MIME CMS Content Type Registry .. 19
   9.5. The SID registry ........................................ 19
   9.6. Media-Type Registry ...................................... 20
       9.6.1. application/voucher-cms+cbor ...................... 20
       9.6.2. application/voucher-cose+cbor .................... 20
   9.7. CoAP Content-Format Registry ............................ 21
10. Acknowledgements ............................................. 21
11. Changelog .................................................... 22
12. References .................................................. 22

Richardson, et al. Expires September 26, 2019 [Page 2]
1. Introduction

Enrollment of new nodes into constrained networks with constrained nodes present unique challenges.

There are bandwidth and code space issues to contend. A solution such as [I-D.ietf-anima-bootstrapping-keyinfra] may be too large in terms of code space or bandwidth required.

This document defines a constrained version of [RFC8366]. Rather than serializing the YANG definition in JSON, it is serialized into CBOR ([RFC7049]).

This document follows a similar, but not identical structure as [RFC8366]. Some sections are left out entirely. Additional sections have been added concerning:

1. Addition of voucher-request specification as defined in [I-D.ietf-anima-bootstrapping-keyinfra],


The CBOR definitions for this constrained voucher format are defined using the mechanism describe in [I-D.ietf-core-yang-cbor] using the SID mechanism explained in [I-D.ietf-core-sid]. As the tooling to convert YANG documents into an list of SID keys is still in its infancy, the table of SID values presented here should be considered normative rather than the output of the pyang tool.

Two methods of signing the resulting CBOR object are described in this document:

1. One is CMS [RFC5652].

2. The other is COSE [RFC8152] signatures.
2. Terminology

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

3. Requirements Language

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant STuPiD implementations.

4. Survey of Voucher Types

[RFC8366] provides for vouchers that assert proximity, that authenticate the registrar and that include different amounts of anti-replay protection.

This document does not make any extensions to the types of vouchers.

Time based vouchers are included in this definition, but given that constrained devices are extremely unlikely to know the correct time, their use is very unlikely. Most users of these constrained vouchers will be online and will use live nonces to provide anti-replay protection.

[RFC8366] defined only the voucher artifact, and not the Voucher Request artifact, which was defined in [I-D.ietf-anima-bootstrapping-keyinfra].

This document defines both a constrained voucher and a constrained voucher-request. They are presented in the order voucher-request, followed by voucher response as this is the time order that they occur.

This document defines both CMS-signed voucher requests and responses, and COSE signed voucher requests and responses. The use of CMS signatures implies the use of PKIX format certificates. The pinned-domain-cert present in such a voucher, is the certificate of the Registrar.

The use of COSE signatures permits the use of both PKIX format certificates, and also raw public keys (RPK). When RPKs are used, the voucher produced by the MASA pins the raw public key of the Registrar: the pinned-domain-subject-public-key-info in such a
voucher, is the raw public key of the Registrar. This is described in the YANG definition for the constrained voucher.

5. Discovery and URI

This section describes the BRSKI extensions to EST-coaps [I-D.ietf-ace-coap-est] to transport the voucher between registrar, proxy and pledge over CoAP. The extensions are targeted to low-resource networks with small packets. Saving header space is important and the EST-coaps URI is shorter than the EST URI.

The presence and location of (path to) the management data are discovered by sending a GET request to "/.well-known/core" including a resource type (RT) parameter with the value "ace.est" [RFC6690]. Upon success, the return payload will contain the root resource of the EST resources. It is up to the implementation to choose its root resource; throughout this document the example root resource /est is used. The example below shows the discovery of the presence and location of voucher resources.

REQ: GET /.well-known/core?rt=ace.est

RES: 2.05 Content
    </est>; rt="ace.est"

The EST-coaps server URIs differ from the EST URI by replacing the scheme https by coaps and by specifying shorter resource path names:

    coaps://www.example.com/est/short-name

Figure 5 in section 3.2.2 of [RFC7030] enumerates the operations and corresponding paths which are supported by EST. Table 1 provides the mapping from the BRSKI extension URI path to the EST-coaps URI path.

<table>
<thead>
<tr>
<th>BRSKI</th>
<th>EST-coaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>/requestvoucher</td>
<td>/rv</td>
</tr>
<tr>
<td>/voucher-status</td>
<td>/vs</td>
</tr>
<tr>
<td>/enrollstatus</td>
<td>/es</td>
</tr>
<tr>
<td>/requestauditlog</td>
<td>/ra</td>
</tr>
</tbody>
</table>

Table 1: BRSKI path to EST-coaps path
/requestvoucher and /enrollstatus are needed between pledge and Registrar.

When discovering the root path for the EST resources, the server MAY return the full resource paths and the used content types. This is useful when multiple content types are specified for EST-coaps server. For example, the following more complete response is possible.

REQ: GET /.well-known/core?rt=ace.est*

RES: 2.05 Content
  </est>; rt="ace.est"
  </est/rv>; rt="ace.est/rv"; ct=50 60 TBD2 TBD3 16
  </est/vs>; rt="ace.est/vs"; ct=50 60
  </est/es>; rt="ace.est/es"; ct=50 60
  </est/ra>; rt="ace.est/ra"; ct=TBD2 TBD3 16

The first line MUST be returned in response to the GET, The following four lines MAY be returned to show the supported Content-Formats. The return of the content-types allows the client to choose the most appropriate one from multiple content types.

Port numbers, not returned in the example, are assumed to be the default numbers 5683 and 5684 for coap and coaps respectively (sections 12.6 and 12.7 of [RFC7252]. Discoverable port numbers MAY be returned in the <href> of the payload.

c=16 stands for the Content-Format "application/cose", and ct=TBD2 stands for Content-Format "application/voucher-cms+cbor, and ct=TBD3 stands for Content-Format "application/voucher-cose+cbor".

Content-Formats TBD2 and TBD3 are defined in this document. The return of the content-formats allows the client to choose the most appropriate one from multiple content formats.

The Content-Format ("application/json") 50 MAY be supported. Content-Formats ("application/cbor") 60, TBD2, TBD3, and 16 MUST be supported.

6. Artifacts

This section describes the abstract (tree) definition as explained in [I-D.ietf-netmod-yang-tree-diagrams] first. This provides a high-level view of the contents of each artifact.
Then the assigned SID values are presented. These have been assigned using the rules in [I-D.ietf-core-yang-cbor], with an allocation that was made via the http://comi.space service.

6.1. Voucher Request artifact

6.1.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with the addition of proximity-registrar-subject-public-key-info, proximity-registrar-cert, and prior-signed-voucher-request. prior-signed-voucher-request is only used between the Registrar and the MASA. proximity-registrar-subject-public-key-info replaces proximity-registrar-cert for the extremely constrained cases.

module: ietf-constrained-voucher-request

grouping voucher-request-constrained-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
        |    +-- proximity-registrar-subject-public-key-info? binary
        |    +-- proximity-registrar-cert? binary
        |    +-- prior-signed-voucher-request? binary

6.1.2. SID values
Base SID value for voucher request: 1001150.

<table>
<thead>
<tr>
<th>SID Assigned to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001167 module ietf-constrained-voucher-request</td>
</tr>
<tr>
<td>1001168 module ietf-restconf</td>
</tr>
<tr>
<td>1001169 module ietf-voucher</td>
</tr>
<tr>
<td>1001170 module ietf-yang-types</td>
</tr>
<tr>
<td>1001171 data /ietf-constrained-voucher-request:voucher</td>
</tr>
<tr>
<td>1001154 data .../ietf-constrained-voucher-request:voucher</td>
</tr>
<tr>
<td>1001155 data .../assertion</td>
</tr>
<tr>
<td>1001156 data .../created-on</td>
</tr>
<tr>
<td>1001157 data .../domain-cert-revocation-checks</td>
</tr>
<tr>
<td>1001158 data .../expires-on</td>
</tr>
<tr>
<td>1001159 data .../idevid-issuer</td>
</tr>
<tr>
<td>1001160 data .../last-renewal-date</td>
</tr>
<tr>
<td>1001161 data .../nonce</td>
</tr>
<tr>
<td>1001162 data .../pinned-domain-cert</td>
</tr>
<tr>
<td>1001165 data .../prior-signed-voucher-request</td>
</tr>
<tr>
<td>1001166 data .../proximity-registrar-cert</td>
</tr>
<tr>
<td>1001163 data .../proximity-registrar-subject-public-key-info</td>
</tr>
<tr>
<td>1001164 data .../serial-number</td>
</tr>
<tr>
<td>1001172 data .../assertion</td>
</tr>
<tr>
<td>1001173 data .../created-on</td>
</tr>
<tr>
<td>1001174 data .../domain-cert-revocation-checks</td>
</tr>
<tr>
<td>1001175 data .../expires-on</td>
</tr>
<tr>
<td>1001176 data .../idevid-issuer</td>
</tr>
<tr>
<td>1001177 data .../last-renewal-date</td>
</tr>
<tr>
<td>1001178 data /ietf-constrained-voucher-request:nonce</td>
</tr>
<tr>
<td>1001179 data .../pinned-domain-cert</td>
</tr>
<tr>
<td>1001180 data .../prior-signed-voucher-request</td>
</tr>
<tr>
<td>1001181 data .../proximity-registrar-cert</td>
</tr>
<tr>
<td>1001182 data .../proximity-registrar-subject-public-key-info</td>
</tr>
<tr>
<td>1001183 data .../serial-number</td>
</tr>
<tr>
<td>1001150 data ietf-constrained-voucher-request</td>
</tr>
<tr>
<td>1001151 data ietf-restconf</td>
</tr>
<tr>
<td>1001152 data ietf-voucher</td>
</tr>
<tr>
<td>1001153 data ietf-yang-types</td>
</tr>
</tbody>
</table>

WARNING, obsolete definitions

6.1.3. YANG Module

In the constrained-voucher-request YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the constrained-
voucher-request module name, all voucher attributes, and the constrained-voucher-request attribute. Two attributes of the voucher are "refined" to be optional.

<CODE BEGINS> file "ietf-constrained-voucher-request@2018-09-01.yang"
module ietf-constrained-voucher-request {  
yang-version 1.1;

namespace
prefix "constrained";

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";
}

organization
   "IETF ANIMA Working Group";

contact
   "WG Web: <http://tools.ietf.org/wg/anima/>
      WG List: <mailto:anima@ietf.org>
      Author: Michael Richardson
             <mailto:mcr+ietf@sandelman.ca>
      Author: Peter van der Stok
             <mailto: consultancy@vanderstok.org>
      Author: Panos Kampanakis
             <mailto: pkampana@cisco.com>";

description
   "This module defines the format for a voucher request, which is produced by a pledge to request a voucher. The voucher-request is sent to the potential owner’s Registrar, which in turn sends the voucher request to the manufacturer or delegate (MASA).

   A voucher is then returned to the pledge, binding the pledge to the owner. This is a constrained version of the voucher-request present in draft-ietf-anima-bootstrap-keyinfra.txt."
This version provides a very restricted subset appropriate for very constrained devices. In particular, it assumes that nonce-ful operation is always required, that expiration dates are rather weak, as no clocks can be assumed, and that the Registrar is identified by a pinned Raw Public Key.

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.

revision "2018-09-01" {
  description
  "Initial version";
  reference
  "RFC XXXX: Voucher Profile for Constrained Devices";
}

c:yang-data voucher-request-constrained-artifact {
  // YANG data template for a voucher.
  uses voucher-request-constrained-grouping;
}

// Grouping defined for future usage
grouping voucher-request-constrained-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 "Base the constrained voucher-request upon the regular one";

      leaf proximity-registrar-subject-public-key-info {
        type binary;
        description
        "The proximity-registrar-subject-public-key-info replaces
the proximit-registrar-cert in constrained uses of the voucher-request.
The proximity-registrar-subject-public-key-info is the Raw Public Key of the Registrar. This field is encoded as specified in RFC7250, section 3.
The ECDSA algorithm MUST be supported.
The EdDSA algorithm as specified in draft-ietf-tls-rfc4492bis-17 SHOULD be supported.
Support for the DSA algorithm is not recommended.
Support for the RSA algorithm is a MAY.
}

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."
}

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-request information when signing a voucher for imprinting so as to minimize the final voucher size.

6.1.4. Example voucher request artifact

Below a CBOR serialization of the constrained-voucher-request is shown in diagnostic CBOR notation. The enum value of the assertion field is calculated to be zero by following the algorithm described in section 9.6.4.2 of [RFC7950].

```
{  
    1001051: {  
        +2 : "2016-10-07T19:31:42Z", / SID= 1001053, created-on /  
        +4 : "2016-10-21T19:31:42Z", / SID= 1001055, expires-on /  
        +1 : 0,                      / SID= 1001052, assertion /  
            "verified" /  
        +10: "JADA123456789",       / SID= 1001061, serial-number /  
        +5 : h’01020D0F’,            / SID= 1001056, idevid-issuer /  
        +15: h’01020D0F’,           / SID=1001066, proximity-registrar-cert/  
            true,                   / SID= 1001054, domain-cert  
            -revocation-checks/  
        +6 : "2017-10-07T19:31:42Z", / SID= 1001057, last-renewal-date /  
        +9 : h’01020D0F’            / SID= 1001060, pinned-domain  
            -subject-public-key-info /  
    }  
}  
```  

6.2. Voucher artifact

The voucher’s primary purpose is to securely assign a pledge to an owner. The voucher informs the pledge which entity it should consider to be its owner.

This document defines a voucher that is a CBOR encoded instance of the YANG module defined in Section 5.3 that has been signed with CMS or with COSE.
6.2.1. Tree Diagram

The following diagram is largely a duplicate of the contents of
[RFC8366], with only the addition of pinned-domain-subject-public-
key-info.

module: ietf-constrained-voucher

grouping voucher-constrained-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
      +-- pinned-domain-subject-public-key-info?  binary

6.2.2. SID values

Base SID value for voucher request: 1001101.

<table>
<thead>
<tr>
<th>SID Assigned to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001115 module ietf-constrained-voucher</td>
</tr>
<tr>
<td>1001116 module ietf-restconf</td>
</tr>
<tr>
<td>1001117 module ietf-voucher</td>
</tr>
<tr>
<td>1001118 module ietf-yang-types</td>
</tr>
<tr>
<td>1001119 data /ietf-constrained-voucher:voucher</td>
</tr>
<tr>
<td>10011110 data .../ietf-constrained-voucher:voucher</td>
</tr>
<tr>
<td>10011111 data .../assertion</td>
</tr>
<tr>
<td>10011112 data .../created-on</td>
</tr>
<tr>
<td>10011113 data .../domain-cert-revocation-checks</td>
</tr>
<tr>
<td>10011114 data .../expires-on</td>
</tr>
<tr>
<td>10011115 data .../idevid-issuer</td>
</tr>
<tr>
<td>10011116 data .../domain-cert-revocation-checks</td>
</tr>
<tr>
<td>10011117 data .../last-renewal-date</td>
</tr>
<tr>
<td>10011118 data .../nonce</td>
</tr>
<tr>
<td>10011119 data .../pinned-domain-cert</td>
</tr>
<tr>
<td>10011120 data .../pinned-domain-subject-public-key-info</td>
</tr>
<tr>
<td>10011121 data .../serial-number</td>
</tr>
</tbody>
</table>
6.2.3. YANG Module

In the constraine-voucher YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the constrained-voucher module name, all voucher attributes, and the constrained-voucher attribute. Two attributes of the voucher are "refined" to be optional.

```xml
<CODE BEGINS> file "ietf-constrained-voucher@2018-09-01.yang"
module ietf-constrained-voucher {
  yang-version 1.1;

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

  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";
  }

  organization
    "IETF ANIMA Working Group";

  contact
    "WG Web:  <http://tools.ietf.org/wg/anima/>"
    "WG List: <mailto:anima@ietf.org>"
    "Author:  Michael Richardson
              <mailto:mcr+ietf@sandelman.ca>"
    "Author:  Peter van der Stok
              <mailto: consultancy@vanderstok.org>"
    "Author:  Panos Kampanakis
              <mailto: pkampana@cisco.com>";

  description
    "This module defines the format for a voucher, which is produced
     by a pledge’s manufacturer or delegate (MASA) to securely assign
     one or more pledges to an ‘owner’, so that the pledges may
     establish a secure connection to the owner’s network
     infrastructure.

     This version provides a very restricted subset appropriate
for very constrained devices.
In particular, it assumes that nonce-ful operation is always required, that expiration dates are rather weak, as no clocks can be assumed, and that the Registrar is identified by a pinned Raw Public Key.


revision "2018-09-01" {
  description
    "Initial version";
  reference
    "RFC XXXX: Voucher Profile for Constrained Devices";
}

rc:yang-data voucher-constrained-artifact {
  // YANG data template for a voucher.
  uses voucher-constrained-grouping;
}

// Grouping defined for future usage
grouping voucher-constrained-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 "Base the constrained voucher upon the regular one";

      leaf pinned-domain-subject-public-key-info {
        type binary;
        description
          "The pinned-domain-subject-public-key-info replaces the pinned-domain-cert in constrained uses of the voucher. The pinned-domain-subject-public-key-info is the Raw Public Key of the Registrar.";
      }
  }
This field is encoded as specified in RFC7250, section 3. The ECDSA algorithm MUST be supported. The EdDSA algorithm as specified in draft-ietf-tls-rfc4492bis-17 SHOULD be supported. Support for the DSA algorithm is not recommended. Support for the RSA algorithm is a MAY.

6.2.4. Example voucher artifacts

Below a the CBOR serialization of the the constrained-voucher and constrained-voucher-request are shown in diagnostic CBOR notation. The enum value of the assertion field is calculated to be zero by following the algorithm described in section 9.6.4.2 of [RFC7950].

```
1001101: {
  +2: "2016-10-07T19:31:42Z", / SID = 1001103, created-on /
  +4: "2016-10-21T19:31:42Z", / SID = 1001105, expires-on /
  +1: 0, / SID = 1001102, assertion /
  / "verified" /
  +12: "JADA123456789", / SID = 1001113, serial-number /
  +5: h'01020D0F', / SID = 1001106, idevid-issuer /
  +8: h'01020D0F', / SID = 1001109, pinned-domain-cert/
  +3: true, / SID = 1001104, domain-cert
  -revocation-checks /
  +6: "2017-10-07T19:31:42Z", / SID = 1001107, last-renewal-date /
  +11: h'01020D0F' / SID = 1001112, proximity
  -registrar-subject-public-key-info /
}
```

6.3. CMS format voucher and voucher-request artifacts

The IETF evolution of PKCS#7 is CMS [RFC5652]. The CMS signed voucher is much like the equivalent voucher defined in [RFC8366]. A different eContentType of TBD1 is used to indicate that the contents are in a different format than in [RFC8366].

The ContentInfo structure contains a payload consisting of the CBOR encoded voucher. The [I-D.ietf-core-yang-cbor] use of delta encoding...
creates a canonical ordering for the keys on the wire. This canonical ordering is not important as there is no expectation that the content will be reproduced during the validation process.

Normally the recipient is the pledge and the signer is the MASA.

[I-D.ietf-anima-bootstrapping-keyinfra] supports both signed and unsigned voucher requests from the pledge to the JRC. In this specification, voucher-request artifact is not signed from the pledge to the registrar. From the JRC to the MASA, the voucher-request artifact MUST be signed by the domain owner key which is requesting ownership.

The considerations of [RFC5652] section 5.1, concerning validating CMS objects which are really PKCS7 objects (cmsVersion=1) applies.

The CMS structure SHOULD also contain all the certificates leading up to and including the signer’s trust anchor certificate known to the recipient. The inclusion of the trust anchor is unusual in many applications, but without it third parties can not accurately audit the transaction.

The CMS structure MAY also contain revocation objects for any intermediate certificate authorities (CAs) between the voucher-issuer and the trust anchor known to the recipient. However, the use of CRLs and other validity mechanisms is discouraged, as the pledge is unlikely to be able to perform online checks, and is unlikely to have a trusted clock source. As described below, the use of short-lived vouchers and/or pledge provided nonce provides a freshness guarantee.

6.3.1. COSE signing

The COSE-Sign1 structure discussed in section 4.2 of [RFC8152]. The CBOR object that carries the body, the signature, and the information about the body and signature is called the COSE_Sign1 structure. It is used when only one signature is used on the body. The signature algorithm is ECSDA with three curves P-256, P-384, and P-512.

Support for EdDSA is encouraged.

Unlike with the CMS structure, the COSE-Sign1 structure does not provide a standard way for the signing keys to be included in the structure. This will not, in general, be a problem for the Pledge, as the key needed to verify the signature MUST be included at manufacturing time.
A problem arises for the Registrar: to verify the voucher, the Registrar must have access to the MASA’s public key. This document does not specify how to transfer the relevant key.

7. Design Considerations

The design considerations for the CBOR encoding of vouchers is much the same as for [RFC8366].

One key difference is that the names of the leaves in the YANG does not have a material effect on the size of the resulting CBOR, as the SID translation process assigns integers to the names.

8. Security Considerations

8.1. Clock Sensitivity

TBD.

8.2. Protect Voucher PKI in HSM

TBD.

8.3. Test Domain Certificate Validity when Signing

TBD.

9. IANA Considerations

9.1. Resource Type Registry

Additions to the sub-registry "CoAP Resource Type", within the "CoRE parameters" registry are specified below. These can be registered either in the Expert Review range (0-255) or IETF Review range (256-9999).

ace.rt.rv needs registration with IANA
ace.rt.vs needs registration with IANA
ace.rt.es needs registration with IANA
ace.rt.ra needs registration with IANA

9.2. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format [RFC3688], the following registration is requested:
9.3. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format defined in [RFC6020], the following registration is requested:

name: ietf-constrained-voucher
prefix: vch
reference: RFC XXXX

name: ietf-constrained-voucher-request
prefix: vch
reference: RFC XXXX

9.4. The SMI Security for S/MIME CMS Content Type Registry

This document registers an OID in the "SMI Security for S/MIME CMS Content Type" registry (1.2.840.113549.1.9.16.1), with the value:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>id-ct-animaCBORVoucher</td>
<td>[ThisRFC]</td>
</tr>
</tbody>
</table>

EDNOTE: should a separate value be used for Voucher Requests?

9.5. The SID registry

The SID range 1001100 was allocated by comi.space to the IETF-CONSTRAINED-VOUCHER yang module.

The SID range 1001150 was allocated by comi.space to the IETF-CONSTRAINED-VOUCHER-REQUEST yang module.

EDNOTE: it is unclear if there is further IANA work required.
9.6. Media-Type Registry

This section registers the 'application/voucher-cms+cbor' media type and the 'application/voucher-cose+cbor' in the "Media Types" registry. These media types are used to indicate that the content is a CBOR voucher either signed with a cms structure or a COSE_Sign1 structure [RFC8152].

9.6.1. application/voucher-cms+cbor

Type name: application
Subtype name: voucher-cms+cbor
Required parameters: none
Optional parameters: none
Encoding considerations: CMS-signed CBOR vouchers are CBOR encoded.
Security considerations: See Security Considerations, Section Interoperability considerations: The format is designed to be broadly interoperable.
Published specification: THIS RFC.
Applications that use this media type: ANIMA, 6tisch, and other zero-touch imprinting systems
Additional information:
  Magic number(s): None
  File extension(s): .vch
  Macintosh file type code(s): none
Person & email address to contact for further information: IETF ANIMA WG
Intended usage: LIMITED
Restrictions on usage: NONE
Author: ANIMA WG
Change controller: IETF
Provisional registration? (standards tree only): NO

9.6.2. application/voucher-cose+cbor
Type name: application
Subtype name: voucher-cose+cbor
Required parameters: none
Optional parameters: cose-type
Encoding considerations: COSE_Sign1 CBOR vouchers are COSE objects signed with one signer.
Security considerations: See Security Considerations, Section Interoperability considerations: The format is designed to be broadly interoperable.
Published specification: THIS RFC.
Applications that use this media type: ANIMA, 6tisch, and other zero-touch imprinting systems
Additional information:
  Magic number(s): None
  File extension(s): .vch
  Macintosh file type code(s): none
Person & email address to contact for further information: IETF ANIMA WG
Intended usage: LIMITED
Restrictions on usage: NONE
Author: ANIMA WG
Change controller: IETF
Provisional registration? (standards tree only): NO

9.7. CoAP Content-Format Registry

Additions to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry are needed for two media types. These can be registered either in the Expert Review range (0-255) or IETF Review range (256-9999).

<table>
<thead>
<tr>
<th>Media type</th>
<th>mime type</th>
<th>Encoding</th>
<th>ID</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/voucher-cms+cbor</td>
<td>-</td>
<td>CBOR</td>
<td>TBD2</td>
<td>[This RFC]</td>
</tr>
<tr>
<td>application/voucher-cose+cbor</td>
<td>&quot;COSE-Sign1&quot;</td>
<td>CBOR</td>
<td>TBD3</td>
<td>[This RFC]</td>
</tr>
</tbody>
</table>

10. Acknowledgements

We are very grateful to Jim Schaad for explaining COSE and CMS choices.

Michel Veillette did extensive work on pyang to extend it to support the SID allocation process, and this document was among the first users.

We are grateful for the suggestions done by Esko Dijk.
11. Changelog

-02

Example of requestvoucher with unsigned application/cbor is added attributes of voucher "refined" to optional CBOR serialization of vouchers improved Discovery port numbers are specified

-01

application/json is optional, application/cbor is compulsory Cms and cose mediatypes are introduced

12. References

12.1. Normative References

[I-D.ietf-ace-cbor-web-token]
   Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)",

[I-D.ietf-ace-coap-est]
   Stok, P., Kampanakis, P., Richardson, M., and S. Raza,
   "EST over secure CoAP (EST-coaps)",
   draft-ietf-ace-coap-est-10 (work in progress), March 2019.

[I-D.ietf-anima-bootstrapping-keyinfra]
   Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen,
   "Bootstrapping Remote Secure Key Infrastructures (BRSKI)",

[I-D.ietf-core-object-security]
   Selander, G., Mattsson, J., Palombini, F., and L. Seitz,
   "Object Security for Constrained RESTful Environments (OSCORE)",
   draft-ietf-core-object-security-16 (work in progress),
   March 2019.

[I-D.ietf-core-sid]
   Veillette, M., Pelov, A., and I. Petrov,
   "YANG Schema Item iDentifier (SID)",
   draft-ietf-core-sid-05 (work in progress), December 2018.


12.2. Informative References


Appendix A. EST messages to EST-coaps

This section extends the examples from Appendix A of [I-D.ietf-ace-coap-est]. The CoAP headers are only worked out for the enrollstatus example.

A.1. enrollstatus

A coaps enrollstatus message can be:

GET coaps://[192.0.2.1:8085]/est/es

The corresponding coap header fields are shown below.
Ver = 1
T = 0 (CON)
Code = 0x01 (0.01 is GET)
Options
  Option1 (Uri-Host)
    Option Delta = 0x3 (option nr = 3)
    Option Length = 0x9
    Option Value = 192.0.2.1
  Option2 (Uri-Port)
    Option Delta = 0x4 (option nr = 4+3=7)
    Option Length = 0x4
    Option Value = 8085
  Option3 (Uri-Path)
    Option Delta = 0x4 (option nr = 7+4= 11)
    Option Length = 0x7
    Option Value = /est/es
Payload = [Empty]

A 2.05 Content response with an unsigned JSON voucher (ct=50) will then be:

  2.05 Content (Content-Format: application/json)
  {payload}

With CoAP fields and payload:

Ver=1
T=2 (ACK)
Code = 0x45 (2.05 Content)
Options
  Option1 (Content-Format)
    Option Delta = 0xC (option nr 12)
    Option Length = 0x2
    Option Value = 0x32 (application/json)
Payload =
  [EDNOTE: put here voucher payload ]

A.2. voucher_status

A coaps voucher_status message can be:

GET coaps://[2001:db8::2:1]:61616]/est/vs

A 2.05 Content response with a non signed CBOR voucher (ct=60) will then be:
2.05 Content (Content-Format: application/cbor)
Payload =
[EDNOTE: put here voucher payload ]

A.3. requestvoucher

Two request-voucher request payloads are possible from pledge to Registrar, a signed one and an unsigned one, as explained in Section 5.2 of [I-D.ietf-anima-bootstrapping-keyinfra].

A.3.1. signed requestvoucher

A coaps signed requestvoucher message from RA to MASA can be:

POST coaps://[2001:db8::2:1]:61616]/est/rv

A 2.04 Changed response returning CBOR voucher signed with a cms structure(ct=TBD2) will then be:

2.04 Changed (Content-Format: application/voucher-cms+cbor)
Payload =
[EDNOTE: put here encrypted voucher payload ]

A.3.2. unsigned requestvoucher

A coaps unsigned requestvoucher message from pledge to Registrar can be:

POST coaps://[2001:db8::2:1]:61616]/est/rv

A 2.04 Changed response returning CBOR voucher (ct=60) will then be:

2.04 Changed (Content-Format: application/cbor)
Payload =
[EDNOTE: put here encrypted voucher payload ]

A.4. requestauditing

A coaps requestauditing message can be:

GET coaps://[2001:db8::2:1]:61616]/est/ra

A 2.05 Content response returning a COSE_Sign1 object (ct=TBD3) will then be:

2.05 Content (Content-Format: application/voucher-cose+cbor)
Payload =
[EDNOTE: put here COSE_Sign1 voucher payload ]
Authors’ Addresses

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Peter van der Stok
vanderstok consultancy

Email: consultancy@vanderstok.org

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com
Abstract

In certain environments, in order for a device to establish any layer three communications, it is necessary for that device to be properly credentialed. This is a relatively easy problem to solve when a device is associated with a human being and has both input and display functions. It is less easy when the human, input, and display functions are not present. To address this case, this memo specifies extensions to the Tunnel Extensible Authentication Protocol (TEAP) method that leverages Bootstrapping Remote Secure Key Infrastructures (BRSKI) in order to provide a credential to a device at layer two. The basis of this work is that a manufacturer will introduce the device and the local deployment through cryptographic means. In this sense the same trust model as BRSKI is used.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 26, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) specifies a means to provision credentials to be used as credentials to operationally access networks. It was designed as a standalone means where some limited access to an IP network is already available. This is not always the case. For example, IEEE 802.11 networks generally require...
authentication prior to any form of address assignment. While it is possible to assign an IP address to a device on some form of an open network, or to accept some sort of default credential to establish initial IP connectivity, the steps that would then follow might well require that the device is placed on a new network, requiring resetting all layer three parameters.

A more natural approach in such cases is to more tightly bind the provisioning of credentials with the authentication mechanism. One such way to do this is to make use of the Extensible Authentication Protocol (EAP) [RFC3748] and the Tunnel Extensible Authentication Protocol (TEAP) method [RFC7170]. Thus we define new TEAP Type-Length-Value (TLV) objects that can be used to transport the BRSKI protocol messages within the context of a TEAP TLS tunnel.

[RFC7170] discusses the notion of provisioning peers. Several different mechanisms are available. Section 3.8 of that document acknowledges the concept of not initially authenticating the outer TLS session so that provisioning may occur. In addition, exchange of multiple TLV messages between client and EAP server permits multiple provisioning steps.

1.1. Terminology

The reader is presumed to be familiar with EAP terminology as stated in [RFC3748]. In addition, the following terms are commonly used in this document.

- **BRSKI**: Bootstrapping Remote Secure Key Infrastructures, as defined in [I-D.ietf-anima-bootstrapping-keyinfra]. The term is also used to refer to the flow described in that document.
- **EST**: Enrollment over Secure Transport, as defined in [RFC7030].
- **Voucher**: a signed JSON object as defined in [RFC8366].

2. TEAP BRISKI Architecture

The TEAP BRISKI architecture is illustrated in Section 3. The device talks to the TEAP server via the Authenticator as per any normal EAP exchange. There is no need for an inner EAP method server, and there is no explicit EAP method type defined for BRISKI.

The architecture illustrated shows the TEAP server and registrar function as being two logically separate entities, however the BRISKI registrar functionality may be integrated into the TEAP server. The device is not explicitly aware of where the registrar functionality is deployed when executing BRISKI inside a TEAP tunnel. Note that the
device may connect directly to the registrar for the purposes of certificate reenrollment, but this happens outside the context to 801.1X and TEAP authentication.

The registrar in turn communicates with the BRSKI MASA service for the purposes of getting signed vouchers. [[TODO: I guess we should mention TEAP server talking to vendor default registrar in the cloud]]

The registrar also communicates with a Certificate Authority in order to issue LDevIDs. The architecture shows the registrar and CA as being two logically separate entities, however the CA may be integrated into the registrar. The device is not explicitly aware of whether the CA and registrar functions are integrated.

3. BRSKI Bootstrap and Enroll Operation

This section summarises the current BRSKI operation. The BRSKI flow assumes the device has an IDevID and has a manufacturer installed trust anchor that can be used to validate the BRSKI voucher. The BRSKI flow compromises several main steps from the perspective of the device:

- Step 1: Device discovers the registrar
- Step 2: Device establishes provisional TLS connection to registrar
- Step 3: Device sends voucher request message and receives signed voucher response
- Step 4: Device validates voucher and validates provisional TLS connection to registrar
- Step 5: Device downloads additional local domain CA information
- Step 6: Device downloads Certificate Signing Request (CSR) attributes
- Step 7: Device does an EST enroll to obtain an LDevID
o Step 8: Device periodically reenrolls via EST to refresh its LDevID

Most of the operational steps require the device, and thus its internal state machine, to automatically complete the next step without being explicitly instructed to do so by the registrar. For example, the registrar does not explicitly tell the device to download additional local domain CA information, or to do an EST enroll to obtain an LDevID.

3.1. Executing BRSKI in a TEAP Tunnel

This section outlines how the main BRSKI steps outlined above map to TEAP, and how BRSKI and enrollment can be accomplished inside a TEAP TLS tunnel. The following new TEAP TLVs are introduced:

- BRSKI-VoucherRequest
- BRSKI-Voucher
- CSR-Attributes

The following steps outline how the above BRSKI flow maps to TEAP.

o Step 1: Device discovers the registrar

When BRSKI is executed in a TEAP tunnel, the device exchanges BRSKI TLVs with the TEAP server. The discovery process for devices is therefore the standard wired or wireless LAN EAP server discovery process. The discovery processes outlined in section 4 of [I-D.ietf-anima-bootstrapping-keyinfra] are not required for initial discovery of the registrar.

o Step 2: Device establishes provisional TLS connection to registrar

The device establishes an outer TEAP tunnel with the TEAP server and does not validate the server certificate. Similarly, at this provisioning stage, the server does not validate the certificate of the device. The device presents its LDevID as its identity certificate if it has a valid LDevID, otherwise it presents its IDevID. Server policy may also be used to control which certificate the device is allowed present, as described in section Section 4.

If the presented credential is sufficient to grant access, the TEAP server can return an EAP-Success immediately. The device may still send a BRSKI-RequestVoucher TLV in response to the EAP-Success if it does not have, but requires, trust anchors for validating the TEAP server certificate.
If the TEAP server requires that the device execute a BRSKI flow, it sends a Request-Action TLV that includes a BRSKI-VoucherRequest TLV. For example, if the device presented its IDevID but the TEAP server requires an LDevID.

The TEAP server may also require the device to reenroll, for example, if the device presented a valid LDevID that is very close to expiration. The server may instruct a device to reenroll by sending a Request-Action TLV that includes a zero byte length PKCS#10 TLV.

- **Step 3:** Device sends voucher request message and receives signed voucher response

The device sends a BRSKI-RequestVoucher TLV to the TEAP server. The TEAP server forwards the RequestVoucher message to the MASA server, and the MASA server replies with a signed voucher. The TEAP server sends a BRSKI-Voucher TLV to the device.

If the MASA server does not issue a signed voucher, the TEAP server sends an EAP-Error TLV with a suitable error code to the device.

For wireless devices in particular, it is important that the MASA server only return a voucher for devices known to be associated with a particular registrar. In this sense, success indicates that the device is on the correct network, while failure indicates the device should try to provision itself within wireless networks (e.g., go to the next SSID).

- **Step 4:** Device validates voucher and validates provisional TLS connection to registrar

The device validates the signed voucher using its manufacturer installed trust anchor, and uses the CA information in the voucher to validate the outer TEAP TLS connection to the TEAP server.

If the device fails to validate the voucher, or fails to validate the outer TEAP TLS connection, then it sends a TEAP-Error TLV indicating failure to the TEAP server.

- **Step 5:** Device downloads additional local domain CA information

On completion of the BRSKI flow, the device SHOULD send a Trusted-Server-Root TLV to the TEAP server in order to discover additional local domain CAs.

- **Step 6:** Device downloads CSR attributes
No later than the completion of step 5, server MUST send a CSR-Attributes TLV to peer server in order to discover the correct fields to include when it enrolls to get an LDevID.

- Step 7: Device does an EST enroll to obtain an LDevID

When executing the BRSKI flow inside a TEAP tunnel, the device does not directly leverage EST when doing its initial enroll. Instead, the device uses the existing TEAP PKCS#10 and PCKS#7 TEAP mechanisms.

Once the BRSKI flow is complete, the device can now send a PKCS#10 TLV to enroll and request an LDevID. If the TEAP server instructed the device to start the BRSKI flow via a Request-Action TLV that includes a BRSKI-RequestVoucher TLV, then the device MUST send a PKCS#10 in order to start the enroll process. The TEAP server will handle the PKCS#10 and ultimately return a PKCS#7 including an LDevID to the device.

If the TEAP server granted the device access on completion of the outer TEAP TLS tunnel in step 2 without sending a Request-Action TLV, the device does not have to send a PKCS#10 to enroll.

At this point, the device is said to be provisioned for local network access, and may authenticate in the future via 802.1X with its newly acquired credentials.

- Step 8: Device periodically reenrolls to refresh its LDevID

When a device’s LDevID is close to expiration, there are two options for re-enrollment in order to obtain a fresh LDevID. As outlined in Step 2 above, the TEAP server may instruct the device to reenroll by sending a Request-Action TLV including a PKCS#10 TLV. If the TEAP server explicitly instructs the device to reenroll via these TLV exchange, then the device MUST send a PKCS#10 to reenroll and request a fresh LDevID.

However, the device SHOULD reenroll if it determines that its LDevID is close to expiration without waiting for explicit instruction from the TEAP server. There are two options to do this.

Option 1: The device reenrolls for a new LDevID directly with the EST CA outside the context of the 802.1X TEAP flow. The device uses the registrar discovery mechanisms outlined in [I-D.ietf-anima-bootstrapping-keyinfra] to discover the registrar and the device sends the EST reenroll messages to the discovered registrar endpoint. No new TEAP TLVs are defined to facilitate discovery of the registrar or EST endpoints inside the context of the TEAP tunnel.
Option 2: When the device is performing a periodic 802.1X authentication using its current LDevID, it reenrolls for a new LDevID by sending a PKCS#10 TLV inside the TEAP TLS tunnel.

4. PKI Certificate Authority Considerations

Careful consideration must be given to PKI certificate authority handling when:

- Establishing the TEAP tunnel
- Establishing trust using BRSKI

These are described in more detail here.

4.1. TEAP Tunnel Establishment

Because this method establishes a client identity, and for purposes of partitioning of responsibility, the peer uses a generic identity string of teap-brsk@TBD1 as its network access identifier (NAI).

The client sends its ClientHello to initiate TLS tunnel establishment. It is possible for the TEAP server to restrict the certificates that the client can use for tunnel establishment by including a list of CA distinguished names in the certificate_authorities field in the CertificateRequest message. Network operators may want to do this in order to restrict network access to clients that have a certificate signed by one of a small set of trusted manufacturer/supplier CAs. If the client has both an IDevID and an LDevID, the client should present the LDevID in preference to its IDevID if allowed by server policy.

In practice, network operators will likely want to onboard devices from a large number of device manufacturers, with each manufacturer using a different root CA when issuing IDevIDs. If the number of different manufacturer root CAs is large, this could result in very large TLS handshake messages. Operators may prefer to include no CAs in the certificate_authorities field thus allowing devices to present IDevIDs signed by any CA when establishing the TEAP tunnel, and instead enforce policy at LDevID enrollment time.

It is recommended that the client validate the certificate presented by the server in the server's Certificate message, but this may not be possible for clients that have not yet provisioned appropriate trust anchors. If the client is in the provisioning phase and has not yet completed a BRSKI flow, it will not have trust anchors installed yet, and thus will not be able to validate the server’s certificate. The client must however note the certificate presented
by the server for (i) inclusion in the BRSKI-RequestVoucher TLV and for (ii) validation once the client has discovered the local domain trust anchors.

If the client does not present a suitable certificate to the server, the server MUST terminate the connection and fail the EAP request.

On establishment of the outer TLS tunnel, the TEAP server will make a policy decision on next steps. Possible policy decisions include:

- Option 1: Server grants client full network access and returns EAP-Success. This will typically happen when the client presents a valid LDevID. Network policy may grant client network access based on IDevID without requiring the device to enroll to obtain an LDevID.

- Option 2: Server requires that client perform a full BRSKI flow, and then enroll to get an LDevID. This will typically happen when the client presents a valid IDevID and network policy requires all clients to have LDevIDs. The server sends a Request-Action TLV that includes a BRSKI-RequestVoucher TLV to the client to instruct it to start the BRSKI flow.

- Option 3: Server requires that the client reenroll to obtain a new LDevID. This could happen when the client presents a valid LDevID that is very close to expiration time, or the server’s policy requires an LDevID update. The server sends an Action-Request TLV including a PKCS#10 TLV to the client to instruct it to reenroll.

4.2. BRSKI Trust Establishment

If the server requires that client perform a full BRSKI flow, it sends a Request-Action TLV that includes a zero byte length BRSKI-RequestVoucher TLV to the client. The client sends a new BRSKI-RequestVoucher TLV to the server, which contains all data specified in [I-D.ietf-anima-bootstrapping-keyinfra] section 5.2. The client includes the server certificate it received in the server’s Certificate message during outer TLS tunnel establishment in the proximity-registrar-cert field. The client signs the request using its IDevID.

The server includes all additional information as required by [I-D.ietf-anima-bootstrapping-keyinfra] section 5.4 and signs the request prior to forwarding to the MASA.

The MASA responds as per [I-D.ietf-anima-bootstrapping-keyinfra] section 5.5. The response may indicate failure and the server should
react accordingly to failures by sending a failure response to the client, and failing the TEAP method.

If the MASA replies with a signed voucher and a successful result, the server then forwards this response to the client in a BRISKI-Voucher TLV.

When the client receives the signed voucher, it validates the signature using its built-in trust anchor list, and extracts the pinned-domain-cert field. The client must use the CA included in the pinned-domain-cert to validate the certificate that was presented by the server when establishing the outer TLS tunnel. If this certificate validation fails, the client must fail the TEAP request and not connect to the network.

[TBD- based on client responses, the registrar sends a status update to the MASA]

5. Channel and Crypto Binding

As the TEAP BRISKI flow does not define or require an inner EAP method, there is no explicit need for exchange of Channel-Binding TLVs between the device and the TEAP server.

The TEAP BRISKI TLVs are expected to occur at the beginning of the TEAP Phase 2 and MUST occur before the final Crypto-Binding TLV. This draft does not exclude the possibility of having other EAP methods occur following the TEAP BRISKI TLVs and as such, the Crypto-Binding TLV process rules as defined in [RFC7170] apply.

6. Protocol Flows

This section outlines protocol flows that map to the 3 server policy options described in section Section 4.1. The protocol flows illustrate a TLS1.2 exchange. Pertinent notes are outlined in the protocol flows.

6.1. TEAP Server Grants Access

In this flow, the server grants access as server policy allows the client to access the network based on the identity certificate that the client presented. This means that either (i) the client has previously completed BRISKI and has presented a valid LDevID or (ii) the client presents an IDevID and network policy allows access based purely on IDevID.
Notes:

(1) If the client has completed the BRSKI flow and has locally significant trust anchors, it must validate the Certificate received from the server. If the client has not yet completed the BRSKI flow, then it provisionally accepts the server Certificate and must validate it later once BRSKI is complete.

(2) The server may include certificate_authorities field in the CertificateRequest message in order to restrict the identity certificates that the device is allowed present.

(3) The device will present its LDevID, if it has one, in preference to its IDevID, if allowed by server policy.

6.2. TEAP Server Instructs Client to Perform BRSKI Flow

In this flow, the server instructs the client to perform a BRSKI flow by exchanging TLVs once the outer TLS tunnel is established.

```
+--------+             +-------------+         +------+
| Client |             | TEAP-Server |         | MASA |
+--------+             +-------------+         +------+

EAP-Request/
Type=Identity
<------------------------>

EAP-Response/
Type=Identity

EAP-Request/
Type=TEAP,          
TEAP Start,         
Authority-ID TLV
```
** At this stage the outer TLS tunnel is established **
** The following message exchanges are for BR SKI **

1. EAP-Response/
   Type=TEAP,
   TLS(ClientHello)

2. EAP-Request/
   Type=TEAP,
   TLS(ServerHello, Certificate, ServerKeyExchange, CertificateRequest, ServerHelloDone)

EAP-Response/
   Type=TEAP,
   TLS(Certificate ClientKeyExchange, CertificateVerify, ChangeCipherSpec, Finished)

EAP-Request/
   Type=TEAP,
   TLS(ChangeCipherSpec, Finished),
   {Crypto-Binding TLV, Result TLV=Success}

EAP-Response/
   Type=TEAP,
   {Crypto-Binding TLV, Result TLV=Success}

EAP-Request/
   Type=TEAP,
   {Request-Action TLV: Status=Failure, Action=Process-TLV, TLV=Request-Voucher,
TLV=Trusted-Server-Root, TLV=CSR-Attributes, TLV=PKCS#10

EAP-Response/
Type=TEAP,
{Request-Voucher TLV}

EAP-Request/
Type=TEAP,
{Voucher TLV}

EAP-Response/
Type=TEAP,
{Trusted-Server-Root TLV}

EAP-Request/
Type=TEAP,
{Trusted-Server-Root TLV}

EAP-Response/
Type=TEAP,
{CSR-Attributes TLV}

EAP-Request/
Type=TEAP,
{CSR-Attributes TLV}

EAP-Response/
Type=TEAP,
{PKCS#10 TLV}

EAP-Request/
Type=TEAP,
{PKCS#7 TLV, Result TLV=Success}
Figure 2: TEAP Server Instructs Client to Perform BRISKI Flow

Notes:

(1) If the client has not yet completed the BRISKI flow, then it provisionally accepts the server certificate and must validate it later once BRISKI is complete.

(2) The server instructs the client to start the BRISKI flow by sending a Request-Action TLV that includes a BRISKI-RequestVoucher TLV. The server also instructs the client to request trust anchors, to request CSR Attributes, and to initiate a PKCS certificate enrolment. As outlined in [RFC7170], the Request-Action TLV is sent after the Crypto-Binding TLV and Result TLV exchange.

(3) The client includes the certificate it received from the server in the RequestVoucher message.

(4) Once the client receives and validates the voucher signed by the MASA, it must verify the certificate it previously received from the server.

(5) As outlined in [RFC7170], the Trusted-Server-Root TLV is exchanged after the Crypto-Binding TLV exchange, and after the client has used the Voucher to authenticate the TEAP server identity.

(6) There is not need for an additional Crypto-Binding TLV exchange as there is no inner EAP method. All BRISKI exchanges are simply TLVs exchanged inside the outer TLS tunnel.

6.3. TEAP Server Instructs Client to Reenroll

In this flow, the server instructs the client to reenroll and get a new LDevID by exchanging TLVs once the outer TLS tunnel is established.
Figure 3: TEAP Server Instructs Client to Reenroll

(1) The server instructs the client to reenroll by sending a Request-Action TLV that includes a PKCS#10 TLV.

6.4. Out of Band Reenroll

This section shows how the device does a reenroll to refresh its LDEvID directly against the registrar outside the context of the TEAP tunnel.
7. TEAP TLV Formats

7.1. BRSKI TLVs

BRSKI defines 3 new TEAP TLVs. The following table indicates whether the TLVs can be included in Request messages from TEAP server to device, or Response messages from device to TEAP server.

<table>
<thead>
<tr>
<th>TLV</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRSKI-VoucherRequest</td>
<td>Response</td>
</tr>
<tr>
<td>BRSKI-Voucher</td>
<td>Request</td>
</tr>
<tr>
<td>CSR-Attributes</td>
<td>Response</td>
</tr>
</tbody>
</table>

These new TLVs are detailed in this section.

7.1.1. BRSKI-RequestVoucher TLV

This TLV is used by the server as part of an Action-Request to request from the peer that it initiate a voucher request. When used in this fashion, the length of this TLV will be set to zero.

It is also used by the peer to initiate the voucher request. When used in this fashion, the length of the TLV will be set to that of the voucher request, as encoded and described in Section 3.3 in [I-D.ietf-anima-bootstrapping-keyinfra].

The M and R bits are always expected to be set to 0.

The server is expected to forward the voucher request to the MASA, and then return a voucher in a BRSKI-Voucher TLV as described below. If it is unable to do so, it returns an TEAP Error TLV with one of the defined errors or the following:
The peer terminates the TEAP connection, but may retry at some later point. The backoff mechanism for such retries should be appropriate for the device. Retries MUST occur no more frequently than once every two (XXX) minutes.

7.1.2. BRSKI-Voucher TLV

This TLV is transmitted from the server to the peer. It contains a signed voucher, as described in [RFC8366].

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|R| TLV=TBD4-Voucher          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              Value...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Upon receiving this TLV the peer will validate the signature of the voucher, using its pre-installed manufacturer trust anchor (LDevID). It MUST also validate the certificate used by the server to establish the TLS connection.

If successful, it installs the new trust anchor contained in the voucher.

Otherwise, the peer transmits an TEAP error TLV with one of the following error messages:

- TBD5-Invalid-Signature: The signature of the voucher signer is invalid.
- TBD6-Invalid-Voucher: The form or content of the voucher is not valid.
- TBD7-Invalid-TLS-Signer: The certificate used for the TLS connection could not be validated.

7.1.3. CSR-Attributes TLV

The server SHALL transmit this TLV to the peer, either along with the BRSKI-Voucher TLV or at any time earlier in a communication. The peer shall include attributes required by the server in any following CSR. The value of this TLV is the base64 encoding described in Section 4.5.2 of [RFC7030].
Again, the M and R values are set to 0. In the case where the client is unable to provide the requested attributes, an TEAP-Error is returned as follows:

TBD9-CSR-Attribute-Fail Unable to supply the requested attributes.

7.2. Existing TEAP TLV Specifications

This section documents allowed usage of existing TEAP TLVs. The definition of the TLV is not changed, however clarifications on allowed values for the TLV fields is documented.

7.2.1. PKCS#10 TLV

[RFC7170] defines the PKCS#10 TLV as follows:

| M | R | TLV Type | Length | PKCS#10 Data... |
+---+---+--------+--------+-----------------+
| 0 | 1 | 2      | 3      |                 |
| 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 |

[RFC7170] does not explicitly allow a Length value of zero.

A Length value of zero is allowed for this TLV when the TEAP server sends a Request-Action TLV with a child PKCS#10 TLV to the client. In this scenario, there is no PKCS#10 Data included in the TLV. Clients MUST NOT send a zero length PKCS#10 TLV to the server.

8. Fragmentation

TLS is expected to provide fragmentation support. Thus EAP-TEAP-BRSKI does not specifically provide any, as it is only expected to be used as an inner method to TEAP.
9. IANA Considerations

The IANA is requested to add entries into the following tables:

The following new TEAP TLVs are defined:

- TBD1-VoucherRequest: Described in this document.
- TBD4-Voucher: Described in this document.
- TBD8-CSR-Attributes: Described in this document.

The following TEAP Error Codes are defined, with their meanings listed here and in previous sections:

- TBD2-MASA-NotAvailable: MASA unavailable
- TBD3-MASA-Refused: MASA refuses to sign the voucher
- TBD5-Invalid-Signature: The signature of the voucher signer is invalid
- TBD6-Invalid-Voucher: The form or content of the voucher is not valid
- TBD7-Invalid-TLS-Signer: The certificate used for the TLS connection could not be validated.
- TBD9-CSR-Attribute-Fail: Unable to supply the requested attributes.

10. Security Considerations

There will be many.

11. Acknowledgments

The authors would like to thank Brian Weis for his assistance, and Alan Dakok for improving language consistency. In addition, with ruthlessly "borrowed" the concept around NAI handling from Tuomas Aura and Mohit Sethi.

12. Informative References


Appendix A. Changes from Earlier Versions

Draft -01: * Add packet descriptions, IANA considerations, smooth out language.

Draft -00:
  o Initial revision

Authors’ Addresses

Eliot Lear
Cisco Systems
Richtistrasse 7
Wallisellen CH-8304
Switzerland

Phone: +41 44 878 9200
Email: lear@cisco.com

Owen Friel
Cisco Systems
170 W. Tasman Dr.
San Jose, CA 95134
United States

Email: ofriel@cisco.com
Abstract

This document discusses the requirement of capability of information distribution among autonomic nodes in autonomic networks. In general, information distribution can be categorized into two different modes: 1) one autonomic node instantly sends information to other nodes in the domain; 2) one autonomic node publishes some information and asynchronously some other interested nodes request the published information. In the former case, information data will be generated and consumed instantly. In the latter case, information data live longer in the network.

These capabilities are basic and fundamental to an autonomous network system (i.e. ANI [I-D.ietf-anima-reference-model]). This document clarifies possible use cases of information distribution in ANI and requirements to ANI so that rich information distribution can be natively supported. Possible options to realize the information distribution function are also briefly discussed.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction ......................................................... 3
2. Terminology ......................................................... 4
3. Requirements of Advanced Information Distribution ................. 4
4. Information Distribution in ANI ...................................... 5
5. Node Behaviors ....................................................... 5
   5.1 Instant Information Distribution ............................... 6
       5.1.1 Instant P2P and Flooding Communications .................. 6
       5.1.2 Instant Selective Flooding Communication ................. 6
   5.2 Asynchronous Information Distribution .......................... 7
       5.2.1 Information Storage ........................................ 7
       5.2.2 Event Queue .............................................. 9
       5.2.3 Interface between IS and EQ Modules ....................... 10
   5.3 Summary ........................................................ 11
   6.1 Un-solicited Synchronization Message (A new GRASP Message) 11
   6.2 Selective Flooding Option ....................................... 11
   6.3 Subscription Objective Option ................................... 12
   6.4 Un_Subscription Objective Option ............................... 12
   6.5 Publishing Objective Option .................................... 13
7. Security Considerations ............................................. 13
1. Introduction

In an autonomic network, autonomic functions (AFs) running on autonomic nodes would exchange information constantly, both for controlling/management signaling and data exchange. This document discusses the information distribution capability of such exchanges between AFs.

According to the number of participants, information distribution can happen with the following scenarios:

1) Point-to-point (P2P) Communication: information are exchanged between two communicating parties from one node to another node.

2) One-to-Many Communication: information exchanges involve an information source and multiple receivers.

The approaches of distributing information could be categorized into two basic models:

1) An instant communication: a sender connects and sends the information data (e.g. control/management signaling, synchronization data and so on) to the receiver(s) immediately.

2) An asynchronous communication: a sender saves the information in the network, may or may not publish the information to the other who is interested in the published information, to which a node asks to retrieve.

The ANI should have provided a generic way to support these various scenarios, rather than assisted by other transport or routing protocols (HTTP, BGP/IGP as bearing protocols etc.). In fact, GRASP already provides part of the capabilities.

In this document, we first analyze requirements of information distribution in autonomic networks (Section 3), and then introduce its relationship to the other modules in ANI (Section 4). After that, the node behaviors and extensions to the existing GRASP are introduced in Section 5 and Section 6, respectively.
2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Requirements of Advanced Information Distribution

If the information exchanged is just short and simple, this can be done instantly. In practice, however, this is not always the case. In the following cases, a mixture of instant and asynchronous communication models is more appropriate.

1) Long Communication Intervals. The time interval of the communication is not necessarily always short and instant. Advanced AFs may rather involve heavy jobs/tasks (e.g. database lookup, authentication etc.) when gearing the network, so the instant mode may introduce unnecessary pending time and become less efficient. If simply using an instant mode, the AF has to wait until the tasks finish and return. A better way is that an AF instantly sends the request but switches to an synchronous mode, once the jobs are finished, AFs will get notified.

2) Common Interest Distribution. As mentioned, some information are common interests among AFs. For example, the network intent is distributed to network nodes enrolled, which is a typical one-to-many scenario. We can also finish the intent distribution by an instant flooding (e.g. via GRASP) to every network nodes across the network domain. Because of network dynamic, however, not every node can be just ready at the moment when the network intent is flooded. Actually, nodes may join in the network sequentially. In this situation, an asynchronous communication model could be a better choice where every (newly joining) node can subscribe the intent information and will get notified if it is ready (or updated).

3) Distributed Coordination. With computing and storage resources on autonomic nodes, alive AFs not only consumes but also generates data information. For example, AFs coordinating with each other as distributed schedulers, responding to service requests and distributing tasks. It is critical for those AFs to make correct decisions based on local information, which might be asymmetric as well. AFs may also need synthetic/aggregated data information (e.g. statistic info, like average values of several AFs, etc.) to make decisions. In these situations, AFs will need an efficient way to form a global view of the network (e.g. about resource consumption, bandwidth and statistics). Obviously, purely relying on instant communication model is inefficient, while a scalable,
common, yet distributed data layer, on which AFs can store and share information in an asynchronous way, should be a better choice.

For ANI, in order to support various communication scenarios, an information distribution module is required, and both instant and asynchronous communication models should be supported.

4. Information Distribution in ANI

This section describes how the information distribution module fits into the ANI including what extensions of GRASP are required [I-D.ietf-anima-grasp].

As the Fig 1 shows, the information distribution module includes three sub-modules, all of which provides APIs for ASAs. Specific behaviors of these modules is described in Section 5. In order to support the modules, the GRASP is also extended, which is described in Section 6.

5. Node Behaviors

ANI is a distributed system, so the information distribution module must be implemented in a distributed way as well. This means that every node participate to contribute. In this section, we discuss how
each autonomic node should behave in order to realize the information distribution module. Node interactions and information data exchange between network nodes are necessary in order to support the instant and asynchronous information distribution, which will be introduced in the follow sections, respectively.

5.1 Instant Information Distribution

In this case, sender(s) and receiver(s) are specified. Information will be directly sent from the sender(s) to the receiver(s). This requires that every node is equipped by some signaling/transport protocols so that they can coordinate with each other and correctly deliver the information.

5.1.1 Instant P2P and Flooding Communications

Current GRASP already provides the capability to support instant P2P and flooding. It is natural to use the GRASP Synchronization message directly for P2P distribution. Furthermore, it is also natural to use the GRASP Flood Synchronization message for 1-to-all distribution.

However, as mentioned in Section 3, in some scenarios one node needs to actively send some information to another. GRASP Synchronization just lacks such capability. An un-solicited synchronization mechanism is needed. A relevant GRASP extension is defined in Section 6.

5.1.2 Instant Selective Flooding Communication

When doing selective flooding, the distributed information needs to contain the criteria for nodes to judge which interfaces should be sent the distributed information and which are not. Specifically, the criteria contain:

- Matching condition: a set of matching rules.
- Matching object: the object that the match condition would be applied to. For example, the matching object could be node itself or its neighbors.
- Action: what behavior the node needs to do when the matching object matches or failed the matching condition. For example, the action could be forwarding or discarding the distributed message.

The criteria information must be include in the message that carries the distributed information from the sender. The receiving node decides the action according to the criteria carried in the message. Still considering the criteria attached with the distributed information, the node behaviors can be:
o When the Matching Object is "Neighbors", then the node matches the relevant information of its neighbors to the Matching Condition. If the node finds one neighbor matches the Matching Condition, then it forwards the distributed message to the neighbor. If not, the node discards forwarding the message to the neighbor.

o When the Matching Object is the node itself, then the node matches the relevant information of its own to the Matching Condition. If the node finds itself matches the Matching Condition, then it forwards the distributed message to its neighbors; if not, the node discards forwarding the message to the neighbors.

An example of selective flooding is briefly described in the Appendix A.

5.2 Asynchronous Information Distribution

Asynchronous information distribution happens in a different way where sender(s) and receiver(s) are normally not immediately specified. Both senders and receivers may come up in an asynchronous way. First of all, this requires that the information can be stored; secondly, it requires an information publication and subscription (Pub/Sub) mechanism (corresponding protocol specification of Pub/Sub is defined in Section 6).

As we sketched in the previous section, in general, each node requires two modules: 1) Information Storage (IS) module and 2) Event Queue (EQ) module in the information distribution module. We introduce details of the two modules in the following sections.

5.2.1 Information Storage

IS module handles how to save and retrieve information for ASAs across the network. The IS module uses a syntax to index information, generating the hash index value (e.g. a key) of the information and mapping the hash index to a certain node in ANI. Note that, this mechanism can use existing solutions. Specifically, storing information in an ANIMA network will be realized in the following steps.

1) ASA-to-IS Negotiation. An ASA calls the API provided by information distribution module (directly supported by IS sub-module) to request to store the information somewhere in the network. Such a request will be checked by the IS module who will be responsible for the request whether such a request is feasible according to criteria such as permitted information size.
2) Destination Node Mapping. The information block will be handled by the IS module in order to calculate/map to a destination node in the network. Since ANIMA network is a peer-to-peer network, a typical way is to use dynamic hash table (DHT) to map information to a unique index identifier. For example, if the size of the information is reasonable, the information block itself can be hashed, otherwise, some meta-data of the information block can be used to generate the mapping.

3) Destination Node Negotiation Request. Negotiation request of storing the information will be sent from the IS module to the IS module on the destination node. The negotiation request contains parameters about the information block from the source IS module. According to the parameters as well as the local available resource, the destination node will feedback the source IS module.

4) Destination Node Negotiation Response. Negotiation response from the destination node is sent back to the source IS module. If the source IS module gets confirmation that the information can be stored, source IS module will prepare to transfer the information block; otherwise, a new destination node must be discovered (i.e. going to step 7).

5) Information Block Transfer. Before sending the information block to the destination node that accepts the request, the IS module of the source node will check if the information block can be afforded by one GRASP message. If so, the information block will be directly sent by calling a GRASP API. Otherwise, bulk data transmission with GRASP will be triggered, where multi-time GRASP message sending will be used so that one information block will be transferred by smaller pieces [I-D.ietf-anima-reference-model].

6) Information Writing. Once the information block (or a smaller block) is received, the IS module of the destination node will store the data block in the local storage, which is accessible.

7) (Optional) New Destination Node Discovery. If the previously selected destination node is not available to store the information block, the source IS module will have to identify a new destination node to start a new negotiation. In this case, the discovery can be done by using discovery GRASP API to identify a new candidate, or more complex mechanisms can be introduced.

Similarly, Getting information from an ANIMA network will be realized in the following steps.

1) ASA-to-IS Request. An ASA accesses the IS module via the APIs exposed by the information distribution module. The key/index of
the interested information will be sent to the IS module. An assumption here is that the key/index should be ready to an ASA before an ASA can ask for the information. This relates to the publishing/subscribing of the information, which are handled by other modules (e.g. Event Queue with Pub/Sub supported by GRASP).

2) Destination Node Mapping. IS module maps the key/index of the requested information to a destination node, and prepares to start to request the information. The mapping here follows the same mechanism when the information is stored.

3) Retrieval Negotiation Request. The source IS module sends a request to the destination node identified in the previous step and asks if such an information object is available.

4) Retrieval Negotiation Response. The destination node checks the key/index of the requested information, and replies to the source IS module. If the information is found and the information block can be afforded within one GRASP message, the information will be sent together with the response to the source IS module.

5) (Optional) New Destination Request. If the information is not found after the source IS module gets the response from the original destination node, the source IS module will have to discover where the location storing the requested information is.

IS module can reuse distributed databases and key value stores like NoSQL, Cassandra, DHT technologies. Storage and retrieval of information are all event-driven responsible by the EQ module.

5.2.2 Event Queue

The main job of Event Queue (EQ) module is to help ASAs to show interests to particular information and notify the occurrences of that in asynchronous communication scenarios. In ANI, information generated on network nodes is labeled as an event identified with an event ID, which is semantically related to the topic of the information. Key features of EQ module are summarized as follows.

1) Event Group: EQ module provides isolated queues for different event groups. If two groups of AFs could have completely different purposes or interests, EQ module allows to create multiple queues where only AFs interested in the same topic will be aware of the corresponding event queue.

2) Event Prioritization: Events do not have to be delivered in the same priority. This corresponds to how much important the event implies. Some of them are more urgent than regular ones.

Prioritization allows AFs to differentiate events (i.e. information) they publish/subscribe.

3) Event Matching: an information consumer has to be identified from the queue in order to deliver the information from the provider. Event matching keeps looking for the subscriptions in the queue to see if there is an exact published event there. Whenever a match is found, it will notify the upper layer to inform the corresponding ASAs who are the information provider and subscriber(s) respectively.

The procedure of how EQ module on every network node works is introduced as follows.

1) Event ID Generation: If information of an ASA is ready, an event ID is generated according to the content of the information. This is also related to how the information is stored/saved by the IS module introduced before. Meanwhile, the type of the event is also specified where it can be of control purpose or user plane data.

2) Priority Specification: According to the type of the event, the ASA may specify its priority to say how this event is wanted to be processed. By considering both aspects, the priority of the event will be determined and ready for enqueuing.

3) Event Enqueue: Given the event ID, event group and its priority, a queue is identified locally if all criteria can be satisfied. If there is such a queue, the event will be simply added into the queue, otherwise a new queue will be created to accommodate such an event.

4) Event Propagation: The published event will be propagated to the other network nodes in the ANIMA domain. A propagation algorithm can be employed to here in order to optimize the propagation efficiency of the updated event queue states.

5) Event Match and Notification: While propagating updated event states, EQ module in parallel keeps matching published events and its interested consumers. Once a match is found, the provider and subscriber(s) will be notified for final information retrieval.

Event contains the address where the information is stored, after a subscriber is notified, it directly retrieves the information from the given location.

5.2.3 Interface between IS and EQ Modules
EQ and IS modules are correlated. When an AF publishes information, not only an publishing event is translated and sent to EQ module, but also the information is indexed and stored simultaneously. Similarly, when an AF subscribes information, not only subscribing event is triggered and sent to EQ module, but also the information will be retrieved by IS module at the same time.

5.3 Summary

In summary, the general requirements for the information distribution module on each autonomic node are two sub-modules handling instant communications and asynchronous communications, respectively. For instant communications, node requirements are simple, in which signaling protocols have to be supported. With minimum efforts, reusing the existing GRASP is possible. For asynchronous communications, information distribution module requires event queue and information storage mechanism to be supported.

6. Protocol Specification (GRASP extension)

There are multiple ways to integrate the information distribution module. The principle we follow is to minimize modifications made to the current ANI.

We consider to use GRASP as an interface to access the information distribution module. The main reason is that the current version of GRASP is already an information distribution module for the cases of P2P and flooding. In the following discussions, we introduce how to complete the missing part.

6.1 Un-solicited Synchronization Message (A new GRASP Message)

In fragmentary CDDL, a Un-solicited Synchronization message follows the pattern:

```
unsolicited_synch-message = [M_UNSOLDSYNCH, session-id, objective]
```

A node MAY actively send a unicast Un-solicited Synchronization message with the Synchronization data, to another node. This MAY be sent to port GRASP_LISTEN_PORT at the destination address, which might be obtained by GRASP Discovery or other possible ways. The synchronization data are in the form of GRASP Option(s) for specific synchronization objective(s).

6.2 Selective Flooding Option

In fragmentary CDDL, the selective flood follows the pattern:
The selective flood option encapsulates a match-condition option which represents the conditions regarding to continue or discontinue flood the current message. For the match-condition option, the Obj1 and Obj2 are to objects that need to be compared. For example, the Obj1 could be the role of the device and Obj2 could be "RSG". The match rules between the two objects could be greater, less than, within, or contain. The match-object represents of which Obj1 belongs to, it could be the device itself or the neighbor(s) intended to be flooded. The action means, when the match rule applies, the current device just continues flood or discontinues.

6.3 Subscription Objective Option

In fragmentary CDDL, a Subscription Objective Option follows the pattern:

```
subscription-objection-option = [SUBSCRIPTION, 2, 2, subobj]
objective-name = SUBSCRIPTION
objective-flags = 2
loop-count = 2
subobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a subscription to a specific object.

6.4 Un_Subscription Objective Option

In fragmentary CDDL, a Un_Subscribe Objective Option follows the pattern:

```
Unsubscribe-objection-option = [UNSUBSCRIB, 2, 2, unsubobj]
objective-name = SUBSCRIPTION
objective-flags = 2
loop-count = 2
unsubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a un-subscription to a
specific object.

6.5 Publishing Objective Option

In fragmentary CDDL, a Publish Objective Option follows the pattern:

```
publish-objection-option = [PUBLISH, 2, 2, pubobj] objective-name
                          = PUBLISH
                          objective-flags = 2
                          loop-count = 2
                          pubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a publish of a specific object data.

Note that extended GRASP messages with new arguments inside here will trigger interactions/actions of the underlying information distribution module introduced in Section 5.

7. Security Considerations

The distribution source authentication could be done at multiple layers:

- Outer layer authentication: the GRASP communication is within ACP (Autonomic Control Plane, [I-D.ietf-anima-autonomic-control-plane]). This is the default GRASP behavior.

- Inner layer authentication: the GRASP communication might not be within a protected channel, then there should be embedded protection in distribution information itself. Public key infrastructure might be involved in this case.

8. IANA Considerations
9. References

9.1 Normative References

[I-D.ietf-anima-grasp]

9.2 Informative References


[I-D.ietf-anima-autonomic-control-plane]

[I-D.ietf-anima-stable-connectivity-10]

[I-D.ietf-anima-reference-model]

[I-D.du-anima-an-intent]

[I-D.ietf-anima-grasp-api]

[I-D.carpenter-anima-grasp-bulk-02]
Carpenter, B., Jiang, S., Liu, B., "Transferring Bulk Data over the GeneRic Autonomic Signaling Protocol (GRASP)", draft-carpenter-anima-grasp-bulk-02 (work in progress),
Appendix A.

GRASP includes flooding criteria together with the delivered information so that every node will process and act according to the criteria specified in the message. An example of extending GRASP with selective criteria can be:

- Matching condition: "Device role=IPRAN_RSG"
- Matching objective: "Neighbors"
- Action: "Forward"

This example means: only distributing the information to the neighbors who are IPRAN_RSG.

Authors’ Addresses

Bing Liu
Huawei Technologies
Q27, Huawei Campus

Liu, et al. Expires September 12, 2019
Appendix A  Real-world Use Cases of Information Distribution

The requirement analysis in Section 3 shows that generally information distribution should be better of as an infrastructure layer module, which provides to upper layer utilizations. In this section, we review some use cases from the real-world where an information distribution module with powerful functions do plays a critical role there.
A.1 Service-Based Architecture (SBA) in 3GPP 5G

In addition to Internet, the telecommunication network (i.e. carrier mobile wireless networks) is another world-wide networking system. The architecture of the upcoming 5G mobile networks from 3GPP has already been defined to follow a service-based architecture (SBA) where any network function (NF) can be dynamically associated with any other NF(s) when needed to compose a network service. Note that one NF can simultaneously associate with multiple other NFs, instead of being physically wired as in the previous generations of mobile networks. NFs communicate with each other over service-based interface (SBI), which is also standardized by 3GPP [3GPP.23.501].

In order to realize an SBA network system, detailed requirements are further defined to specify how NFs should interact with each other with information exchange over the SBI. We now list three requirements that are related to information distribution here.

1) NF Pub/Sub: Any NF should be able to expose its service status to the network and any NF should be able to subscribe the service status of an NF and get notified if the status is available. An concrete example is that a session management function (SMF) can subscribe the REGISTER notification from an access management function (AMF) if there is a new user entity trying to access the mobile network [3GPP.23.502].

2) Network Exposure Function (NEF): A particular network function that is required to manage the event exposure and distributions. In specific, SBA requires such a functionality to register network events from the other NFs (e.g. AMF, SMF and so on), classify the events and properly handle event distributions accordingly in terms of different criteria (e.g. priorities) [3GPP.23.502].

3) Network Repository Function (NRF): A particular network function where all service status information is stored for the whole network. An SBA network system requires all NFs to be stateless so as to improve the resilience as well as agility of providing network services. Therefore, the information of the available NFs and the service status generated by those NFs will be globally stored in NRF as a repository of the system. This clearly implies storage capability that keeps the information in the network and provides those information when needed. A concrete example is that whenever a new NF comes up, it first of all registers itself at NRF with its profile. When a network service requires a certain NF, it first inquires NRF to retrieve the availability information and decides whether or not there is an available NF or a new NF must be instantiated [3GPP.23.502].
(Note: 3GPP CT might finally adopt HTTP2.0/JSON to be the protocol communicating between NFs, but autonomic networks can also load HTTP2.0 with in ACP.)

A.2 Vehicle-to-Everything

Carrier networks On-boarding services of vertical industries are also one of some blooming topics that are heavily discussed. Connected car is clearly one of the important scenarios interested in automotive manufacturers, carriers and vendors. 5G Automotive Alliance - an industry collaboration organization defines many promising use cases where services from car industry should be supported by the 5G mobile network. Here we list two examples as follows [5GAA.use.cases].

1) Software/Firmware Update: Car manufacturers expect that the software/firmware of their car products can be remotely updated/upgraded via 5G network in future, instead of onsite visiting their 4S stores/dealers offline as nowadays. This requires the network to provide a mechanism for vehicles to receive the latest software updates during a certain period of time. In order to run such a service for a car manufacturer, the network shall not be just like a network pipe anymore. Instead, information data have to be stored in the network, and delivered in a publishing/subscribing fashion. For example, the latest release of a software will be first distributed and stored at the access edges of the mobile network, after that, the updates can be pushed by the car manufacturer or pulled by the car owner as needed.

2) Real-time HD Maps: Autonomous driving clearly requires much finer details of road maps. Finer details not only include the details of just static road and streets, but also real-time information on the road as well as the driving area for both local urgent situations and intelligent driving scheduling. This asks for situational awareness at critical road segments in cases of changing road conditions. Clearly, a huge amount of traffic data that are real-time collected will have to be stored and shared across the network. This clearly requires the storage capability, data synchronization and event notifications in urgent cases from the network, which are still missing at the infrastructure layer.

A.3 Summary

Through the general analysis and the concrete examples from the real-world, we realize that the ways information are exchanged in the coming new scenarios are not just short and instant anymore. More advanced as well as diverse information distribution capabilities are required and should be generically supported from the infrastructure
layer. Upper layer applications (e.g. ASAs in ANIMA) access and utilize such a unified mechanism for their own services.
Abstract

This document details the mechanism used for initial enrollment using a smartphone of a BRSKI Registrar system.

There are two key differences in assumption from [I-D.ietf-anima-bootstrapping-keyinfra]: that the intended registrar has Internet, and that the Pledge has no user-interface.

This variation on BRSKI is intended to be used in the situation where the registrar device is new out of the box and is the intended gateway to the Internet (such as a home gateway), but has not yet been configured. This work is also intended as a transition to the Wi-Fi Alliance work on the Device Provisioning Protocol (DPP).

Status of This Memo

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

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

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

This Internet-Draft will expire on September 12, 2019.
1. Introduction .................................................. 3
  1.1. Intermittent Device connectivity ..................... 4
  1.2. Additional Motivation ................................. 4
2. Terminology .................................................... 5
  2.1. History and Origin of the name ........................ 6
3. Requirements Language .......................................... 6
4. Assumptions and Required Setup ................................ 6
5. Protocol Overview .............................................. 7
  5.1. Scan the QR code ........................................ 7
  5.2. Enroll with the manufacturer .......................... 7
  5.3. Connect to BRISKI join network ........................ 7
  5.4. Connect to Adolescent Registrar (AR) .................. 8
  5.5. Pledge Requests Voucher-Request from the Adolescent
        Registrar .................................................. 8
  5.6. AR processing of voucher-request, request. .......... 9
       5.6.1. Additions to Voucher-Request .................... 9
  5.7. Smartphone validates connection ........................ 9
  5.8. Smart-Phone connects to MASA .......................... 10
  5.9. MASA processing ........................................ 10
  5.10. Smartpledge processing of voucher ..................... 10
  5.11. Adolescent Registrar (AR) receives voucher .......... 11
  5.12. Adolescent Registrar (AR) grows up .................... 11
  5.13. Smartphone enrolls .................................... 11
  5.14. Validation of connection ................................ 12
6. Protocol Details ............................................... 12
  6.1. Quick Response Code (QR code) ......................... 12
       6.1.1. The Smarkaklink Attribute ........................ 13
       6.1.2. Link-Layer Address Attribute ...................... 13
       6.1.3. ESSID Name Attribute ................................ 14
  6.2. Enrollment using EST .................................... 14
7. Smart Pledge enrollment with manufacturer .................. 14
1. Introduction

The problem of bootstrapping a new device is described at length in [I-D.ietf-anima-bootstrapping-keyinfra] (aka BRSKI). The problem that BRSKI solves is the case of a smart, properly configured network with a minimum of network connectivity (or previously pre- provisioned with nonceless vouchers), and a relatively stupid new device (the Pledge), which lacks a user interface.

The BRSKI problem is one of trust: how does the new device trust that it has found the correct network to join, and how does the new network become convinced that the new device is a device that is intended to join. BRSKI solves the problem well for the case where the network is well connected and can easily talk to the device’s Manufacturer Authorized Signing Authority (MASA), while providing appropriate proxy mechanisms to enable the new pledge to communicate it’s proximity assertion to the MASA as well.

This document is about a variation of the problem: when the new device being introduce has no network connectivity, but a new device is intended to serve as the Registrar for the network. This new device is likely a home (or small office) gateway, and until it is properly configured there will be no direct network connectivity.

There are a number of protocols that permit an ISP to consider a new router brought into a home to be a new pledge to the ISPs’ network, and for that new device to integrated into the ISP’s (autonomic) network. BRSKI can be used itself, and there are ways to use the Broadband Form’s TR-069 to bootstrap the device in this way. This document is not about the situation where the router device is
intended to belong to the ISP, but about the situation where the home
user intends to own and control the device.

1.1. Intermittent Device connectivity

There is an additional variation which this variation solves: the
case where there is one or more devices in a place with no immediate
connectivity to a Registrar. An example of this could be a new home
construction where a furnace, thermostat or other control systems
need to be introduced to each other. If a registrar exists it will
have no Internet connectivity (as above), until the home becomes
owned by the first owner. There might never be a registrar though.

The basement case is important because the assumption is that the
installer may have poor or no LTE connectivity in that location.
The installer will have to exit the basement, perhaps even return to
their truck, in order to have network connectivity for their
provisioning device (a smartphone equivalent).

1.2. Additional Motivation

The Wi-Fi Alliance has released the Device Provisioning Protocol
[dpp]. The specification is available only via "free" registration.
The specification relies on being able to send and receive 802.11
Public Action frames, as well as Generic Advertisement Service (GAS)
Public Action frames. Access to send new layer-2 frames is generally
restricted in most smartphone operating systems (iOS, Android). At
present there are no known public APIs that a generic application
writer could use, and therefore the smart-phone side of the DPP can
only be implemented at present by the vendors of those operating
systems.

As both dominant vendors have competing proprietary mechanisms, it is
unclear if generic applications will be produced soon. It is
probably impractical for a vendor an a smart-appliance to
independantly produce an application that can do proper DPP in 2019.
As one of the common goals of this document and DPP is that there
need not be an application-per-device only one DPP application need
exist. Until such time as such an application becomes universal it
is a goal of this document to lay the groundwork for a transition to
full use of DPP by leveraging the QR code infrastructure that DPP
depends upon.

In addition to the above concern, DPP is primary concerned about
provisioning WiFi credentials to devices. DPP can provision access
points themselves, but it lacks any kind of manufacturer integration.
BRISKI provides this integration, and therefore an audit trail history
for the device.
The smarkaklink enrollment process described in this document is about securely initializing the administrative connection with a device that is the WiFi Access Point.

2. Terminology

The following terminology is copied from [I-D.ietf-anima-bootstrapping-keyinfra]

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 at the factory.

IDevID: a manufacturer signed keypair (different from the QRkey) which is generated at the factory. This is the 802.1AR artifact which is mandated by [I-D.ietf-anima-bootstrapping-keyinfra].

The following new terminology has been added

smarkaphone: The prospective administrator device, usually a smartphone equipped with a QR capable camera, wifi and 3G connectivity.

adolescent router (AR): a home router or device containing a registrar. The device does not yet have network connectivity, and has no administrator. It is considered not a "baby" device in the same way that the pledge is, but it is not yet an adult. A better term would be welcome.

SelfDevID: a public/private key pair generated by the smartpledge, formed into a self-signed PKIX certificate. The private key part remains always on the smartpledge, but like other secondary device keys, should be encrypted for backup purposes. (EDNOTE: any references to Apple or Android APIs/specifications here?)

QRkey: a unique, raw ECDSA or EdDSA key pair generated in (or for) the adolescent router at the factory, and stored in the configuration portion of the firmware. The public portion is printed in a QRcode. This key is not formed into a certificate of any kind.

smarkaklink: the name of this protocol.
adolescent router (AR): a home router or device containing a registrar. The device does not yet have network connectivity, and has no administrator.

2.1. History and Origin of the name

This document was originally called the "smartpledge" variation of BRSKI. This name was intended to indicate that the variation is one where the BRSKI role of Pledge is taken on by the smartphone device. While the end-goal is to have the smartphone enrolled into a PKI hosted by the fully-grown Router, the activities of each device do not map into the BRSKI roles at the beginning. In fact, they are reversed with the Adolescent Router being the Pledge. Review of this document suggested that removing the word pledge would help. The new name "smarkaklink" is intended to sound like the sound that two (wine, beer) glasses make after a toast is made.

3. Requirements Language

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant STuPiD implementations.

4. Assumptions and Required Setup

The first assumption is that intended device owner is active and is present. The device owner has a smart-phone that is capable of using Wi-Fi or being wired into the adolescent router (AR).

The smartpledge application generates a self-signed certificate with public/private keypair that it knows. It may generate a unique certificate for each manufacturer. This certificate is called the SelfDevID.

The second assumption is that the device has a QR code printed on the outside of the unit, and/or provided with the packaging/documentation. The QR code is as specified in section 5.3 of [dpp], with the additions specified in Section 6.1.

The third assumption is that the AR, at manufacturing time, has the anchor for it’s MASA (same assumption as for BRSKI pledge’s). In addition, like the BRSKI pledge, the AR has an IDevID certificate (and associated private key) signed by the manufacturer.
The fourth assumption is that the key in the "K:" attribute
Section 6.1 is a different public key pair. It MUST be different
from the key used in the IDevID. This key is called the DPP-Keypair.

5. Protocol Overview

This is the overview of the process. {EDNOTE: there are many details
here that belong in the next section. The goal in this section is to
consisely explain the interaction among the components. Clearly this
text currently fails in that regard}

5.1. Scan the QR code

The operator of the smartphone invokes the smarkaklink application,
and scans the QR code on the AR. The smartphone learns the ESSID,
Public-Key, mac-address, smarkaklink URL, and link-local address of
the AR.

5.2. Enroll with the manufacturer

The smartphone uses it’s 3G, or other WiFi internet access to connect
to the manufacturer with TLS. The manufacturer is identified with
the smarthpledge URL.

The operator of the smartphone may need to move to another location
to get connectivity. It is desireable that an operator be able to
scan many QR codes before moving, performing this operation in a
batch. There may be multiple devices from the same manufacturer, and
the smarkaklink application SHOULD enroll with the manufacturer a
single time for all devices.

The smartphone does an HTTP POST to the provided URL using it’s
generated certificate as it’s ClientCertificate. As described in
Section 7, the manufacturer MAY respond with a 302 result code, and
have the end user go through a web browser based process to enroll.
After that process, a redirection will occur using OAUTH2.

The result should finally be a 201 result code, and at that URL is a
new certificate signed by the manufacturer.

5.3. Connect to BRKSI join network

The application then reconnects the Wi-Fi interface of the smartphone
to the ESSID of the AR. This involves normal 802.11 station
attachment. The given ESSID explicitely has no WPA or other security
required on it.
There will be no DHCPv4 on this network. This simplifies the operation of the devices that are enrolling, but it also makes the network uninteresting to other random users that may stumble upon the open ESSID.

A IPv6 Router Solicitation may elicit an answer (confirming the device is there), but it is acceptable for there to be no prefix information. An IPv6 Neighbour Discovery is done for the IPv6 Link-Local address of the AR. Receipt of an answer confirms that the ESSID is correct and present.

(XXX - not using GRASP here. Could use GRASP, but QR code is better)

5.4. Connect to Adolescent Registrar (AR)

The smarkaklink application then makes a direct (no proxy) TLS connection to port 8443 (!XXX!) of the AR, on the IPv6 Link-Local address given. This is as in section 5.1 of [I-D.ietf-anima-bootstrapping-keyinfra]. The smartphone uses its SelfDevID as the TLS ClientCertificate, as the smartphone and smarkaklink will not have a manufacturer signed IDevID.

Additionally, the AR will use its IDevID certificate as the ServerCertificate of the TLS connection. As with other BRSKI IDevID, it will have a MASA URL extension, as described in [I-D.ietf-anima-bootstrapping-keyinfra] section 2.3.2.

The Adolescent Registrar acts in the role of pledge!

5.5. Pledge Requests Voucher-Request from the Adolescent Registrar

The smartphone generates a random nonce _SPnonce_. To this is added SOMETHING-that-is-time-unique, to create a _voucher-request challenge_. This is placed in the voucher-challenge-nonce field.

Using the public-key of the AR that was scanned from the QR code, the smartphone encrypts the challenge using CMS (or COSE?).

NOTE: DPP has a round with the SHA256 of the device’s key to make sure that the correct device has been chosen. The TLS connection effectively provides the same privacy that the Bx keys provided.

The resulting object is POST’ed to the new BRSKI endpoint:

/.well-known/est/requestvoucherrequest

[or should it be named: /.well-known/est/requestvouchercertificate]
5.6. AR processing of voucher-request, request.

The AR processes this POST. First it uses the private key that is associated with it’s QR printed public key to decrypt the voucher-request challenge. Included in this challenge is a nonce, and also the link-local address of the smartphone.

The AR SHOULD verify that the link-local address matches the originating address of the connection on which the request is received.

The AR then forms a voucher-request identically to as described in section 5.2 of [I-D.ietf-anima-bootstrapping-keyinfra]. Note that the AR uses it’s IDevID to sign the voucher-request. This is the same key used to terminate the TLS connection.

Note: It MUST be different from the public key printed in the QR code.

In addition to the randomly generated nonce that the AR generates to place in the the voucher-request, into the nonce field, it also includes the _SPnonce_ in a new _voucher-challenge-nonce_ field. (EDNOTE: hash of nonce?)

This voucher-request is then _returned_ during the POST operation to the smartphone. (This is in constrast that in ANIMA the voucher-request is sent by the device to the Registrar, or the MASA)

5.6.1. Additions to Voucher-Request

QUESTION: should the _voucher-challenge-nonce_ be provided directly in the voucher-request, or should only a hash of the nonce be used? The nonce is otherwise not disclosed, and a MITM on the initial TLS connection would get to see the nonce. A hash of the nonce validates the nonce as easily.

5.7. Smartphone validates connection

The smartphone then examines the resulting voucher-request. The smartphone validates that the voucher-request is signed by the same public key as was seen in the TLS ServerCertificate.

The smartphone then examines the contents of the voucher-request, and looks for the _voucher-challenge-nonce_. As this nonce was encrypted to the AR, the only way that the resulting nonce could be correct is if the correct private key was present on the AR to decrypt it.
Successful verification of the _voucher-challenge-nonce_ (or the hash of it, see below) results in the smartphone moving its end of the connection from provisional to validated.

5.8. Smart-Phone connects to MASA

The smarkaklink application running on the smartphone then examines the MASA URL provided in the TLS ServerCertificate of the AR. The smarkaklink application then connects to that URL using its 3G/LTE connection, taking on the temporary role of Registrar.

A wrapped voucher-request is formed by the smartphone in the same way as described in section 5.4 of [I-D.ietf-anima-bootstrapping-keyinfra]. The prior-signed-voucher-request is filled in with the voucher-request that was created by the AR in the previous step.

The proximity-registrar-cert of the wrapped voucher-request is set to be the SelfDevID certificate of the smartphone. The voucher-request is to be signed by the SelfDevID.

The voucher-request is POST’ed to the MASA using the same URL that is used for Registrar/MASA operation:

/.well-known/est/requestvoucher

5.9. MASA processing

The MASA processing occurs as specified in section 5.5 of [I-D.ietf-anima-bootstrapping-keyinfra] as before. The MASA MUST also copy the _voucher-challenge-nonce_ into the resulting voucher.

5.10. Smartpledge processing of voucher

The smartphone will receive a voucher that contains its IDevID as the _pinned-domain-cert_, and the _voucher-challenge-nonce_ that it created will also be present. The smartphone SHOULD verify the signature on the artifact, but may be unable to validate that the certificate used has a relationship to the TLS ServerCertificate used by the MASA. (This limitation exists in ANIMA as well).

The smartphone will then POST the resulting voucher to the AR using the URL

/.well-known/est/voucher

If an existing TLS connection is still available, it MAY be reused.
If a TLS session-resumption ticket (see [RFC8446] section 2.2 for TLS 1.3, and [RFC5077] for TLS 1.2) has been obtained, it SHOULD be used if the TLS connection needs to be rebuilt. This is particularly useful in the disconnected use case explained in Section 1.1.

5.11. Adolescent Registrar (AR) receives voucher

When the AR receives the voucher, it validates that it is signed by its manufacturer. This process is the same as section 5.5.1 of [I-D.ietf-anima-bootstrapping-keyinfra].

Again note that the AR is acting in the role of a pledge.

Inside the voucher, the pinned-domain-cert is examined. It should match the TLS ClientCertificate that the smartphone used to connect. This is the SelfDevID.

At this point the AR has validated the identity of the smartphone, and the AR moves it’s end of the connection from provisional to validated.

5.12. Adolescent Registrar (AR) grows up

The roles are now changed.

If necessary, the AR generates a new key pair as it’s Domain CA key. It MAY generate intermediate CA certificates and a separate Registrar certificate, but this is discouraged for home network use.

The AR is now considered a full registrar. The AR now takes on the role of Registrar.

5.13. Smartphone enrolls

At this stage of the smarkaklink protocol, the typical BRSKI exchange is over. A Secure Transport has been established between the smartphone and the fully-grown AR. The smartphone now takes on the role of secured pledge, or EST client.

The smartphone MUST now request the full list of CA Certificates, as per [RFC7030] section 4.1. As the Registrar’s CA certificate has just been generated, the smartphone has no other way of knowing it.

The smartphone MUST now also generate a CSR request as per [I-D.ietf-anima-bootstrapping-keyinfra] section 5.8.3. The smartpledge MAY reuse the SelfDevID key pair for this purpose. (XXX – maybe there are good reasons not to reuse?)
The Registrar SHOULD grant administrator privileges to the smartphone via the certificate that is issued. This may be done via special attributes in the issued certificate, or it may pin the certificate in a database. Which method to use is a local matter.

The TLS/EST connection MUST remain open at this point. This is connection one.

5.14. Validation of connection

The smartphone MUST now open a new HTTPS connection to the Registrar (AR), using it’s newly issued certificate. (XXX should this be on a different IP, or a different port? If so, how is this indicated?)

The smartphone MUST validate that the new connection’s TLS Server certificate can be validated by the Registrar’s new CA certificate.

The registrar MUST validate that the smartphone’s ClientCertificate is validated by the Registrar’s CA. The smartphone SHOULD perform a POST operation on this new connection to the [I-D.ietf-anima-bootstrapping-keyinfra] Enrollment Status Telemetry mechanism, see section 5.8.3.

Upon success, the original TLS/EST connection (one) MAY now be closed.

Should the validations above fail, then the original EST connection MUST be used to GET a value from the

/.well-known/est/enrollstatus

from the Registrar. The contents of this value SHOULD then be sent to the MASA, using a POST to the enrollstatus, and including the reply from the AR in a new attribute, "adolescent-registrar-reason".

6. Protocol Details

6.1. Quick Response Code (QR code)

Section 5.3 of [dpp] describes the contents for an [iso18004] image. It specifies content that starts with DPP:, and the contains a series of semi-colon (;) delimited section with a single letter and colon. This markup is terminated with a double semi-colon.

Although no amending formula is defined in DPP 1.0, this document is defining two extensions. This requires amending the ABNF from section 5.2.1 as follows:
While the ABNF defined in the [dpp] document assumes a specific order (C:, M:, I:, K:), this specification relaxes this so that the tags can come in any order. However, in order to make interoperation with future DPP-only clients as seamless as possible, the extensions suggested here are placed at the end of the list. This is consistent with the Postel Principle.

It is intended that parts of this protocol could be performed by an actual DPP implementation, should it become possible to implement DPP using current smartphone operating systems in an unprivileged way.

6.1.1. The Smarkaklink Attribute

The _smarkaklink_ attribute indicates that the device is capable of the protocol specified in this document. The contents of the smarkaklink attribute contains part or all of an IRI which identifies the manufacturer of the device.

It SHOULD contain the _iauthority_ of an IRI as specified in section 2.2 of [RFC3987]. The scheme is implicitly "https://", with an ipath of "/.well-known/est/smarkaklink". This implicit form exists to save bytes in the QR code.

If the string contains any "/" characters, then it is not an _iauthority_, but an entire IRI. This takes many more characters, but is useful in a variety of debugging situations, and also provides for new innovations.

Short URLs are important to fit into typical QR code space.

6.1.2. Link-Layer Address Attribute

The _llv6-addr_ attribute is optional. When present, it specifies the IPv6 Link-Local address at which the adolescent router is listening. If not specified, then the link-local address may be formed according to the historical (privacy-violating) process described in [RFC4291] Appendix A. The _llv6-addr_ attribute is present so that devices that have implemented [RFC7217] stable addresses can express that address clearly.
6.1.3. ESSID Name Attribute

The _essid_ attribute provides the name of the 802.11 network on which the enrollment will occur. If this attribute is absent, then it defaults to "BRSKI".

6.2. Enrollment using EST

TBD

7. Smart Pledge enrollment with manufacturer

While it is assumed that there will be many makers of Smarkaklink applications, a goal of this specification is to eliminate the need for an "app" per device, providing onboarding mechanism for a variety of devices from a single app.

Given the secondary goal of a transition to use of Device Provisioning Protocol (DPP), the smarkaklink application may have to be provided as part of the smart phone system, as a system service. This is due to the need to send/receive wifi management frames from DPP. As such each vendor of a smart device will need to produce a smarkaklink app, and it will be impossible for the vendor of the Registrar device (or other DPP capable IoT device) to provide an app on their own.

Having stated this goal, it is understood that initially the app may well come from the manufacturer of the Registrar, but this protocol is designed on the assumption that there is no such vertical integration.

So, there can be no initial relationship between the Smart Pledge and the manufacturer of the Registrar.

But, in a traditional [I-D.ietf-anima-bootstrapping-keyinfra] scenario the pledge would have been provided with an IDevID at manufacturing time. While an IDevID could have been built-in to the SmartPledge "app", such a key would not be private if it was built-in. A key could be generated by the app upon installation. It could be self-signed, it could be signed by the maker of the app, or it could be signed by another party.

- a self-signed certificate is just a container for a public key. For the purposes of the trust relationship with the Registrar, it would be sufficient.
- a certificate signed by the maker of the app (or the maker of the smart-phone) would carry no specific trust beyond what a self-
signed certificate would have. Any linking in the certificate to a network expressable identity (such as layer-2 address) would simply be a privacy violation.

- a certificate signed by another party would similarly have little additional relevance, unless the third party is the manufacturer of the Registrar!

The smarkaklink enrollment process uses a combination of the first and third choice. The involvement of the manufacturer at this step affords an opportunity to do sales-channel integration with the manufacturer. The manufacturer can associate an account with the user using a wide variety of OAUTH2 [RFC6749] processes. In addition, based upon the URL provided the manufacturer can do redirection along a value-added reseller process. For instance, the manufacturer of a home router could redirect the pledge to the ISP that resold the router.

While [RFC7030] describes a Certificate Signing Request in order to have a certificate assigned, the actual contents of the certificate are not interesting at all, and the process of attempting to come up with a meaningful contents tends to cause more interoperability issues than having nothing.

The Smarkaklink takes the _smartpledge attribute_ from the QR code, forming a URL as described above. An HTTPS POST is performed to this URL, with the JSON body of:

```json
{
  "mac" : <mac-address>
}
```

The HTTPS POST MUST be performed with freshly created self-signed certificate. If the smarkaklink application has previously communicated with this URL, it MAY skip this step and use a previously returned certificate. Doing so has a privacy implication discussed below, but is appropriate when enrolling many devices from the same manufacturer into the same network.

The smarkaklink client should be prepared for three cases:

- A certificate is immediately returned.
- A 201 status code is returned, and Location: header is provided. A GET request to that location will retrieve the certificate.
- A 302 redirection occurs with some initiation of an OAUTH2 process to establish some additional authorization.
Any other error (4xx and 5xx) are typically unrecoverable errors.

In the third case, the 302 response SHOULD take the smarkaklink operator to the given URL in an interactive browser. The operator SHOULD be given access to their normal set of cookies and third-party logins such that they can use appropriate third party (Google, Facebook, Github, Live.com, etc.) logins to help validate the operator as a real person, and not a malware. Such logins are optional, and it is a manufacturer choice as to what integrations they want to make.

After the OAUTH2 process, the SmartPledge will be redirected back to the MASA and a 201 status code will be returned when successful as above.

7.1. minimal Smart Pledge enrollment

A manufacturer who has not built-in any restrictions on the identity that the smarkaklink uses, MAY return the same self-signed certificate that the smartpledge used to connect with.

8. Threat Analysis

The following attacks have been considered.

8.1. Wrong Administrator

Neighbours with similar setups wind up managing each other’s network (by mistake).

8.2. Rogue Administrator

Uninitialized networks can be adopted by ‘wardrivers’ who search for networks that have no administrator.

8.3. Attack from Internal device

A compromised device inside the home can be used by an attack to take control of the home router.

8.4. Attack from camera enabled robot

A robot (such as a home vacuum cleaner) could be compromised, and then used by an attacker to observe and/or scan the router QRcode.
8.5. Attack from manipulator enabled robot

A robot (for instance, a toy) could be compromised, and then used by
an attacker to push the WPA and/or factory reset button on the
router.

9. Security Considerations

XXX: Go through the list of attacks above, and explain how each has
been mitigated.

Go through the list of concerns in ANIMA and EST-RFC7030 and indicate
if there are additional concerns, or if a concern does not apply.

10. IANA Considerations

TBD.

11. Acknowledgements

This work was supported by the Canadian Internet Registration

12. References

12.1. Normative References

<https://www.wi-fi.org/downloads-registered-guest/Device_P
_v0_0_23_0.zip/31255>.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason,
S., and K. Watsen, "Bootstrapping Remote Secure Key
Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-

[iso18004]
"Information technology --- Automatic identification and
data capture techniques --- Bar code symbology --- QR
Codes (ISO/IEC 18004:2015)", n.d.,
<https://github.com/yansikeim/QR-Code/blob/master/
12.2. Informative References


Appendix A. Resulting DPP QR code specification

This is a merge of the additions from section Section 6.1 and section 5.2.1 of [dpp]:

Richardson, et al. Expires September 12, 2019 [Page 18]
Internet-Draft                 Smarkaklink                    March 2019

dpp-qr = "DPP:" [channel-list ";"]
        [";" llv6-addr ] [";" mudurl ]
        [";" smarkaklink ] [";" essid ] ";;"
llv6-addr = "L:" 8*hex-octet
essid = "E:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
smarkaklink = "S:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
mudurl = "D:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
pkex-bootstrap-info = [information]
channel-list = "C:" class-and-channels *("","" class-and-channels)
class-and-channels = class "/" channel *(","" channel)
class = 1*3DIGIT
channel = 1*3DIGIT
mac = "M:" 6hex-octet ; MAC address
hex-octet = 2HEXDIG
information = "I:" *(%x20-3A / %x3C-7E) ; semicolon not allowed
public-key = "K:" *PKCHAR
    ; DER of ASN.1 SubjectPublicKeyInfo encoded in
    ; "base64" as per [14]
PKCHAR = ALPHA / DIGIT / %x2b / %x2f / %x3d
llv6-addr = "L:" 8*hex-octet
essid = "E:" *(%x21-3A / %x3C-7E) ; semicolon not allowed
smartpledge = "S:" *(%x21-3A / %x3C-7E) ; semicolon not allowed

Appendix B. Swagger.IO definition of API

This is a work-in-progress definition of the smarkaklink to MASA API
in the form of Swagger.IO format:

---

swagger: "2.0"
info:
    description: |
        The smartpledge API is described in detail in
draft-richardson-anima-smartpledge. This API is
a variation of BRSKI (draft-ietf-anima-bootstrapping-keyinfra)
which provides an initial bootstrap of the
Secure Home Gateway registrar.
version: 1.0.0
title: Secure Home Gateway secure enrollment API (smartpledge-BRSKI)
contact:
    email: securehomegateway@cira.ca
license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
host: virtserver.swaggerhub.com
basePath: /CIRALabs/smartpledge/1.0.0
tags:
    - name: est
description: Enrollment over Secure Transport
schemes:
- https
paths:
  /voucherrequest:
    get:
      tags: [developers]
      summary: searches inventory
      description: |
        By passing in the appropriate options, you can search for
        available inventory in the system
      operationId: searchInventory
      produces:
        - application/json
      parameters:
        - name: searchString
          in: query
          description: |
            pass an optional search string for looking up inventory
          required: false
          type: string
        - name: skip
          in: query
          description: number of records to skip for pagination
          required: false
          type: integer
          minimum: 0
          format: int32
        - name: limit
          in: query
          description: maximum number of records to return
          required: false
          type: integer
          maximum: 50
          minimum: 0
          format: int32
      responses:
        200:
          description: search results matching criteria
          schema:
            type: array
            items:
              $ref: '#/definitions/InventoryItem'
        400:
          description: bad input parameter
- admins
summary: adds an inventory item
description: Adds an item to the system
operationId: addInventory
consumes:
- application/json
produces:
- application/json
parameters:
- in: body
  name: inventoryItem
  description: Inventory item to add
  required: false
  schema:
    $ref: '#/definitions/InventoryItem'
responses:
  201:
    description: item created
  400:
    description: invalid input, object invalid
  409:
    description: an existing item already exists
definitions:
  InventoryItem:
    type: object
    required:
      - id
      - manufacturer
      - name
      - releaseDate
    properties:
      id:
        type: string
        format: uuid
        example: d290f1ee-6c54-4b01-90e6-d701748f0851
      name:
        type: string
        example: Widget Adapter
      releaseDate:
        type: string
        format: date-time
        example: 2016-08-29T09:33:00Z
      manufacturer:
        $ref: '#/definitions/Manufacturer'
  Manufacturer:
    required:
      - name
    properties:
name:
  type: string
  example: ACME Corporation

homePage:
  type: string
  format: url
  example: https://www.acme-corp.com

phone:
  type: string
  example: 408-867-5309

Authors’ Addresses

  Michael Richardson
  Sandelman Software Works

  Email: mcr+ietf@sandelman.ca

  Jacques Latour
  CIRA Labs

  Email: Jacques.Latour@cira.ca

  Faud Khan
  Twelve Dot Systems

  Email: faud.khan@twelvedot.com

  Abhishek Joshi
  Twelve Dot Systems

  Email: abhishek.joshi@twelvedot.com
Constrained Join Proxy for Bootstrapping Protocols

draft-vanderstok-anima-constrained-join-proxy-01

Abstract

This document defines a protocol to securely assign a pledge to an owner, using an intermediary node between pledge and owner. This intermediary node is known as a "constrained-join-proxy".

This document extends the work of [ietf-anima-bootstrapping-keyinfra] by replacing the Circuit-proxy by a stateless constrained join-proxy, that transports routing information.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 11, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
1. Introduction

Enrollment of new nodes into constrained networks with constrained nodes present is described in [I-D.ietf-anima-bootstrapping-keyinfra] and makes use of Enrollment over Secure Transport (EST) [RFC7030]. The specified solutions use https and may be too large in terms of code space or bandwidth required. Constrained devices in constrained networks [RFC7228] typically implement the IPv6 over Low-Power Wireless personal Area Networks (6LoWPAN) [RFC4944] and Constrained Application Protocol (CoAP) [RFC7252].
CoAP has chosen Datagram Transport Layer Security (DTLS) [RFC6347] as the preferred security protocol for authenticity and confidentiality of the messages. A constrained version of EST, using Coap and DTLS, is described in [I-D.ietf-ace-coap-est].

DTLS is a client-server protocol relying on the underlying IP layer to perform the routing between the DTLS Client and the DTLS Server. However, the new "joining" device will not be IP routable until it is authenticated to the network. A new "joining" device can only initially use a link-local IPv6 address to communicate with a neighbour node using neighbour discovery [RFC6775] until it receives the necessary network configuration parameters. However, before the device can receive these configuration parameters, it needs to authenticate itself to the network to which it connects. In [I-D.ietf-anima-bootstrapping-keyinfra] Enrolment over Secure Transport (EST) [RFC7030] is used to authenticate the joining device. However, IPv6 routing is necessary to establish a connection between joining device and the EST server.

This document specifies a Join-proxy and protocol to act as intermediary between joining device and EST server to establish a connection between joining device and EST server.

This document is very much inspired by text published earlier in [I-D.kumar-dice-dtls-relay].

2. Terminology

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

3. Requirements Language

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant STuPiD implementations.

4. Join Proxy functionality

As depicted in the Figure 1, the joining Device, or pledge (P), is more than one hop away from the EST server (E) and not yet authenticated into the network. At this stage, it can only communicate one-hop to its nearest neighbour, the Join proxy (J) using their link-local IPv6 addresses. However, the Pledge (P) needs
to communicate with end-to-end security with a Registrar hosting the EST server (E) to authenticate and get the relevant system/network parameters. If the Pledge (P) initiates a DTLS connection to the EST server whose IP address has been pre-configured, then the packets are dropped at the Join Proxy (J) since the Pledge (P) is not yet admitted to the network or there is no IP routability to Pledge (P) for any returned messages.

![Diagram](https://via.placeholder.com/150)

**Figure 1: multi-hop enrolment.**

Furthermore, the Pledge (P) may wish to establish a secure connection to the EST server (E) in the network assuming appropriate credentials are exchanged out-of-band, e.g. a hash of the Pledge (P)’s raw public key could be provided to the EST server (E). However, the Pledge (P) is unaware of the IP address of the EST-server (E) to initiate a DTLS connection and perform authentication with.

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

5. Join Proxy specification

The Join Proxy can operate in two modes:

- Statefull mode
- Stateless mode

In the statefull mode two configuration are envisaged:

- Join Proxy knows EST Server address
5.1. Statefull Join Proxy

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

Assume the Pledge knows the address of the EST server. The message is transmitted to the EST Server as if it originated from the joining node, by replacing the IP address and port of the Pledge to the DTLS IP address of the proxy and a randomly chosen port. The DTLS message itself is not modified. Consequently, the Join Proxy must track the ongoing DTLS connections based on the following 4-tuple stored locally:

- Pledge link-local IP address (IP_C)
- Pledge source port (p_C)
- EST Server IP address (IP_S)
- EST Server source port (p_R)

The EST Server communicates with the Join Proxy as if it were communicating with the Pledge, without any modification required to the DTLS messages. On receiving a DTLS message from the EST Server, the Join Proxy looks up its locally stored 4-tuple array to identify to which Pledge (if multiple exist) the message belongs. The DTLS message’s destination address and port are replaced with the link-local address and port of the corresponding Pledge and the DTLS message is then forwarded to the Pledge. The Join Proxy does not modify the DTLS packets and therefore the normal processing and security of DTLS is unaffected.

In Figure 2 the various steps of the process are shown where the EST Server address is known to the Pledge:

<table>
<thead>
<tr>
<th>EST Client (P)</th>
<th>Join-Proxy (J)</th>
<th>EST Server (E)</th>
<th>Message Src_IP:port</th>
<th>Dst_IP:port</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ClientHello--&gt;</td>
<td>--ClientHello--&gt;</td>
<td>IP_C:p_C</td>
<td>IP_S:5684</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP_R:p_R</td>
<td>IP_S:5684</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;--ServerHello--</td>
<td>IP_S:5684</td>
<td>IP_R:p_R</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;--ServerHello--</td>
<td>IP_S:5684</td>
<td>IP_C:p_C</td>
<td></td>
</tr>
</tbody>
</table>
Assume that the pledge does not know the IP address of the EST Server it needs to contact. In that situation, the Join Proxy can be configured with the IP address of a default EST Server that an EST client needs to contact. The EST client initiates its request as if the Join Proxy is the intended EST Server. The Join Proxy changes the IP packet (without modifying the DTLS message) as in the previous case by modifying both the source and destination addresses to forward the message to the intended EST Server. The Join Proxy keeps a similar 4-tuple array to enable translation of the DTLS messages received from the EST Server and forwards it to the EST Client. In Figure 3 the various steps of the message flow are shown:

<table>
<thead>
<tr>
<th>EST Client (P)</th>
<th>Join Proxy (J)</th>
<th>EST Server (E)</th>
<th>Message Src_IP:port</th>
<th>Dst_IP:port</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;--ClientHello--</td>
<td>&lt;--ServerHello--</td>
<td>IP_S:5684</td>
<td>IP_C:p_C</td>
<td>IP_Rb:p_Rb</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP_S:5684</td>
<td>IP_C:p_C</td>
<td>IP_Ra:5684</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP_S:5684</td>
<td>IP_C:p_C</td>
<td>IP_Ra:5684</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP_S:5684</td>
<td>IP_C:p_C</td>
<td>IP_Ra:5684</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP_S:5684</td>
<td>IP_C:p_C</td>
<td>IP_Ra:5684</td>
</tr>
</tbody>
</table>

IP_C:p_C = Link-local IP address and port of EST Client
IP_S:5684 = IP address and coaps port of EST Server
IP_R:p_R = IP address and port of Join Proxy

Figure 2: constrained statefull joining message flow with EST server address known to Join Proxy.
5.2. Stateless Join Proxy

The Join-proxy is stateless to minimize the requirements on the constrained Join-proxy device.

When a joining device as a client attempts a DTLS connection to the EST server, it uses its link-local IP address as its IP source address. This message is transmitted one-hop to a neighbour node. Under normal circumstances, this message would be dropped at the neighbour node since the joining device is not yet IP routable or it is not yet authenticated to send messages through the network. However, if the neighbour device has the Join Proxy functionality enabled, it routes the DTLS message to a specific EST Server. Additional security mechanisms need to exist to prevent this routing functionality being used by rogue nodes to bypass any network authentication procedures.

If an untrusted DTLS Client that can only use link-local addressing wants to contact a trusted end-point EST Server, it sends the DTLS message to the Join Proxy. The Join Proxy extends this message into a new type of message called Join ProxY (JPY) message and sends it on to the EST server. The JPY message payload consists of two parts:

- Header (H) field: consisting of the source link-local address and port of the Pledge (P), and
- Contents (C) field: containing the original DTLS message.

On receiving the JPY message, the EST Server retrieves the two parts. The EST Server transiently stores the Header field information. The EST server uses the Contents field to execute the EST server functionality. However, when the EST Server replies, it also extends its DTLS message with the header field in a JPY message and sends it back to the Join Proxy. The Header contains the original source link-local address and port of the DTLS Client from the transient state stored earlier (which can now be discarded) and the Contents field contains the DTLS message.
On receiving the JPY message, the Join Proxy retrieves the two parts. It uses the Header field to route the DTLS message retrieved from the Contents field to the Pledge.

The Figure 4 depicts the message flow diagram when the EST Server end-point address is known only to the Join Proxy:

<table>
<thead>
<tr>
<th>EST Client (P)</th>
<th>Join Proxy (J)</th>
<th>EST server (E)</th>
<th>Message</th>
<th>Src_IP:port</th>
<th>Dst_IP:port</th>
</tr>
</thead>
<tbody>
<tr>
<td>--ClientHello--</td>
<td></td>
<td></td>
<td></td>
<td>IP_C:p_C</td>
<td>IP_Ra:5684</td>
</tr>
<tr>
<td></td>
<td>--JPY[H(IP_C:p_C),--&gt;</td>
<td>IP_Rb:p_Rb</td>
<td>IP_S:5684</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(ClientHello)]</td>
<td></td>
<td></td>
<td>IP_S:5684</td>
<td>IP_Rb:p_Rb</td>
</tr>
<tr>
<td>--ServerHello--</td>
<td></td>
<td></td>
<td></td>
<td>IP_Ra:5684</td>
<td>IP_C:p_C</td>
</tr>
<tr>
<td></td>
<td>--JPY[H(IP_C:p_C),--</td>
<td>IP_S:5684</td>
<td>IP_Rb:p_Rb</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(ServerHello)]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--Finished--&gt;</td>
<td></td>
<td></td>
<td></td>
<td>IP_C:p_C</td>
<td>IP_Ra:5684</td>
</tr>
<tr>
<td></td>
<td>--JPY[H(IP_C:p_C),--&gt;</td>
<td>IP_Rb:p_Rb</td>
<td>IP_S:5684</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>C(Finished)]</td>
<td></td>
<td></td>
<td>IP_S:5684</td>
<td>IP_Rb:p_Rb</td>
</tr>
<tr>
<td>--Finished--</td>
<td></td>
<td></td>
<td></td>
<td>IP_Ra:5684</td>
<td>IP_C:p_C</td>
</tr>
</tbody>
</table>

IP_C:p_C = Link-local IP address and port of the Pledge
IP_S:5684 = IP address and coaps port of EST Server
IP_Ra:5684 = Link-local IP address and coaps port of Join Proxy
IP_Rb:p_Rb = IP address(can be same as IP_Ra) and port of Join Proxy

JPY[H(),C()] = Join ProxY message with header H and content C

Figure 4: constrained stateless joining message flow.

5.3. Stateless Message structure

The JPY message is constructed as a payload with media-type application/multipart-core specified in [I-D.ietf-core-multipart-ct]. Header and Contents fields use different media formats:

1. header field: application/CBOR containing a CBOR array [RFC7049] with the pledge IPv6 Link Local address as a 16-byte binary value, the pledge’s UDP port number, if different from 5684, as a CBOR integer, and the proxy’s ifindex or other identifier for the
physical port on which the pledge is connected. Header is not DTLS encrypted.

2. Content field: Any of the media types specified in
   [I-D.ietf-ace-coap-est] and [I-D.ietf-anima-constrained-voucher] dependent on the function that is requested:

   * application/pkcs7-mime; smime-type=server-generated-key
   * application/pkcs7-mime; smime-type=certs-only
   * application/voucher-cms+cbor
   * application/voucher-cose+cbor
   * application/pkcs8
   * application/csrattrs
   * application/pkcs10
   * application/pkix-cert

   Examples are shown in Appendix A. The content fields are DTLS encrypted.

6. Comparison of stateless and statefull modes

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

   +-----------------+---------------------------------+------------------------+
   | Properties       | Stateful mode                   | Stateless mode         |
   +-----------------+---------------------------------+------------------------+
   | State Information| The Proxy needs additional      | No information is      |
   |                  | storage to maintain mapping     | maintained by the Join |
   |                  | of the Pledge’s address         | Proxy                  |
   |                  | with the port number being used |                        |
   |                  | to communicate with the Server. |                        |
   +-----------------+---------------------------------+------------------------+
   | Packet size      | The size of the forwarded       | Size of the forwarded   |
   |                  | message is the same as the      | message is bigger than  |
   |                  | original message.               | the original, it includes|
   |                  |                                  | additional source and   |
   |                  |                                  | destination addresses.  |
   +-----------------+---------------------------------+------------------------+
   | Specification    | The additional functionality      | New JPY message to      |
   | complexity       | the Proxy to maintain state     | encapsulate DTLS message|
   |                  | information, and modify the     | The Server and the proxy|
   |                  | source and destination          | have to understand the  |
   |                  | addresses of the DTLS           | JPY message in order    |
   |                  | handshake messages              | to process it.          |
   +-----------------+---------------------------------+------------------------+
7. Discovery

It is assumed that Join-Proxy seamlessly provides a coaps connection between Pledge and coaps EST-server. An additional Registrar is needed to connect the Pledge to an http EST server, see section 8 of [I-D.ietf-ace-coap-est].

The Discovery of the coaps EST server by the Join Proxy follows section 6 of [I-D.ietf-ace-coap-est]. The discovery of the Join-Proxy by the Pledge is an extension to the discovery described in section 4 of [I-D.ietf-anima-bootstrapping-keyinfra]. In particular this section replaces section 4.2 of [I-D.ietf-anima-bootstrapping-keyinfra]. Three discovery cases are discussed: coap discovery, 6tisch discovery and GRASP discovery.

7.1. GRASP discovery

In the context of autonomous networks, discovery takes place via the GRASP protocol as described in [I-D.ietf-anima-bootstrapping-keyinfra]. The port number is.

EDNote: to be specified further

7.2. 6tisch discovery

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

7.3. Coaps discovery

In the context of a coap network without Autonomous Network support, discovery follows the standard coap policy. The Pledge can discover a Join-Proxy by sending a link-local multicast message to ALL CoAP Nodes with address FF02::FD. Multiple or no nodes may respond. The handling of multiple responses and the absence of responses follow section 4 of [I-D.ietf-anima-bootstrapping-keyinfra].
The presence and location of (path to) the join-proxy resource are discovered by sending a GET request to "/.well-known/core" including a resource type (rt) parameter with the value "brski-proxy" [RFC6690]. Upon success, the return payload will contain the root resource of the Join-Proxy resources. It is up to the implementation to choose its root resource; throughout this document the example root resource /est is used. The example below shows the discovery of the presence and location of join-proxy resources.

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

RES: 2.05 Content
   </est>; rt="brski-proxy";ct=62

Port numbers, not returned in the example, are assumed to be the default numbers 5683 and 5684 for coap and coaps respectively (sections 12.6 and 12.7 of [RFC7252]). Discoverable port numbers MAY be returned in the <href> of the payload.

8. Security Considerations

It should be noted here that the contents of the CBOR map are not protected, but that the communication is between the Proxy and a known registrar (a connected UDP socket), and that messages from other origins are ignored.

9. IANA Considerations

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

9.1. Resource Type registry

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

rt="brski-proxy". This EST resource is used to query and return the supported EST resource of a join-proxy placed between Pledge and EST server.
10. Acknowledgements

Many thanks for the comments by Brian Carpenter.

11. Contributors

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

12. Changelog

12.1. 00 to 01

- Added Contributors section
- Adapted content-formats to est-coaps formats
- Aligned examples with est-coaps examples
- Added statefull Proxy to stateless proxy

12.2. 00 to 00

- added payload examples in appendix
- discovery for three cases: AN, 6tisch and coaps

13. References

13.1. Normative References

[I-D.ietf-6tisch-enrollment-enhanced-beacon]

[I-D.ietf-ace-coap-est]
Stok, P., Kampanakis, P., Kumar, S., Richardson, M., Furuhe, M., and S. Raza, "EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-est-00 (work in progress), February 2018.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key

Richardson, et al. Expires September 11, 2019
13.2. Informative References

[I-D.kumar-dice-dtls-relay]

[RFC4944]

[RFC6690]
Appendix A. Stateless Proxy payload examples

Examples are extensions of two examples shown in [I-D.ietf-ace-coap-est].

EDNote:
provisional stake holder examples to be improved and corrected.

A.1. cacerts

The request from Join-Proxy to EST-server looks like:

Get coaps://192.0.2.1/est/crts
(Accept: 62)
(Content-format: 62)
payload =
82 array(2)
18 3C unsigned(60)
The response will then be

2.05 Content
(Content-format: 62)
Payload =

```plaintext
83 # array(3)
18 3C # unsigned(60)
83 # array(3)
69 # text(9)
464538303A3A414238 # "FE80::AB8"
19 237D # unsigned(9085)
65 # text(5)
6964656E74 # "ident"
82 # array(2)
19 0119 # unsigned(281)
59 027F # bytes(639)
3082027b06092a864886f7d0d010702a082026c308202680201013100300b
06092a864886f7d0d010701a082024a3082024a308201f0a0030201020209
09189bcd9c9924bb300a06082a8648ce3d0403023067310b30090630055
040613025553310b300906305504080c024341310b300906305504070c02
4c4131143012060355040a0c0b4578161706c5204966331163014063
55040b0c0d6365727469669636174696f6e3110300e06035504030c752
6f674204341301e170d3139303130373130343034315a1630339303130
323130333034315a3067310b3009060355040613025553310b3009063055
04080c024341310b300906305504070c024c4131143012060355040a0c0b
4578616d706c6520496e636572746966796c55652ac
2cbb7c54a50a7c7d7bc722da6c85ca538209fddbf104c9a38184308181
301d0603515d0e041604142495e816ef6ffcaaf356ce4adffe33cf24bb2
a8301f0603515d20418301680142495e816ef6ffcaaf356ce4adffe33cf
492abba8300f0603551d130101ff040503003011ff0300e603551d0f0101
ff04003020106301e0603551d110417305811363657246966796406578
616d760652e63666d30a06082a8648ce3d04302034800030450210da
0e37c96f154c32ec0b4af52d46f3b7ecc9687dd267bcecc368f7b7f135327
2f022047a28ae5c7306163b3c3834bab3c103f74307059c089aa00ac870
cd13b902ca1003100
```
A.2. serverkeygen

The request from Join-Proxy to EST-server looks like:

Get coaps://192.0.2.1/est/skg
(Accept: 62)
(Content-Format: 62)
Payload =

83 # array(3)
18 3C # unsigned(60)
83 # array(3)
69 # text(9)
464538303A3A414238 # "FE80::AB8"
19 237D # text(9085)
65 # text(5)
6964656E74 # "ident"
82 # array(2)
19 011E # unsigned(286)
58 D2 # bytes(210)
3081cf3078020100301631143012060355040a0c0b736b767206578616d70
6c653059301306072a8648ce3d020106082a8648ce3d03010703420041b
b8c1117896f98e4506c03d70f838ea97e9d6552c8460c5852c5
1dd89a61370a2843760fc859799d78c33f3c18646e304f1717f8123f1a28
4cc99fa00000a06082a8648ce3d04030204470030440220387cd4e9cf62
8d4af77f92ebced48909d9d14dca86cd2757dd14c359cd69f618020202f24
5e828c77754378b66660a4977f113cacda0cc7bad7d1474a7fd155d090d

The response will then be

2.05 Content
(Content-format: 62)
Payload =

84 # array(4)
18 3C # unsigned(60)
83 # array(3)
69 # text(9)
464538303A3A414238 # "FE80::AB8"
19 237D # unsigned(9085)
65 # text(5)
6964656E74 # "ident"
82 # array(2)
19 011E # unsigned(286)
58 8A # bytes(138)
3081cf3078020100301631143012060355040a0c0b736b767206578616d70
6c653059301306072a8648ce3d020106082a8648ce3d03010703420041b
b8c1117896f98e4506c03d70f838ea97e9d6552c8460c5852c5
1dd89a61370a2843760fc859799d78c33f3c18646e304f1717f8123f1a28
4cc99fa00000a06082a8648ce3d04030204470030440220387cd4e9cf62
8d4af77f92ebced48909d9d14dca86cd2757dd14c359cd69f618020202f24
5e828c77754378b66660a4977f113cacda0cc7bad7d1474a7fd155d090d

Authors' Addresses

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

Peter van der Stok
vanderstok consultancy
Email: consultancy@vanderstok.org

Panos Kampanakis
Cisco Systems
Email: pkampana@cisco.com