

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 26 January 2021

B. E. Carpenter
Univ. of Auckland
L. Ciavaglia
Nokia
S. Jiang
Huawei Technologies Co., Ltd
P. Peloso
Nokia
25 July 2020

Guidelines for Autonomic Service Agents
draft-carpenter-anima-asa-guidelines-09

Abstract

This document proposes guidelines for the design of Autonomic Service Agents for autonomic networks, as a contribution to describing an autonomic ecosystem. It is based on the Autonomic Network Infrastructure outlined in the ANIMA reference model, using the Autonomic Control Plane and the Generic Autonomic Signaling 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 26 January 2021.

Copyright Notice

Copyright (c) 2020 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	2
2. Logical Structure of an Autonomic Service Agent	4
3. Interaction with the Autonomic Networking Infrastructure	5
3.1. Interaction with the security mechanisms	5
3.2. Interaction with the Autonomic Control Plane	5
3.3. Interaction with GRASP and its API	6
3.4. Interaction with policy mechanism	7
4. Interaction with Non-Autonomic Components	7
5. Design of GRASP Objectives	8
6. Life Cycle	9
6.1. Installation phase	9
6.1.1. Installation phase inputs and outputs	10
6.2. Instantiation phase	11
6.2.1. Operator's goal	11
6.2.2. Instantiation phase inputs and outputs	12
6.2.3. Instantiation phase requirements	12
6.3. Operation phase	13
7. Coordination between Autonomic Functions	14
8. Coordination with Traditional Management Functions	14
9. Robustness	14
10. Security Considerations	15
11. IANA Considerations	16
12. Acknowledgements	16
13. References	16
13.1. Normative References	16
13.2. Informative References	17
Appendix A. Change log [RFC Editor: Please remove]	19
Appendix B. Example Logic Flows	20
Authors' Addresses	25

1. Introduction

This document proposes guidelines for the design of Autonomic Service Agents (ASAs) in the context of an Autonomic Network (AN) based on the Autonomic Network Infrastructure (ANI) outlined in the ANIMA reference model [I-D.ietf-anima-reference-model]. This infrastructure makes use of the Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane] and the Generic Autonomic Signaling Protocol (GRASP) [I-D.ietf-anima-grasp]. This document is a contribution to the description of an autonomic ecosystem, recognizing that a deployable autonomic network needs more than just

ACP and GRASP implementations. It must achieve management goals that a Network Operations Center (NOC) cannot achieve manually, including at least a library of ASAs and corresponding GRASP objective definitions. There must also be tools to deploy and oversee ASAs, and integration with existing operational mechanisms [RFC8368]. However, this document focuses on the design of ASAs, with some reference to implementation and operational aspects.

There is a considerable literature about autonomic agents with a variety of proposals about how they should be characterized. Some examples are [DeMola06], [Huebscher08], [Movahedi12] and [GANAI13]. However, for the present document, the basic definitions and goals for autonomic networking given in [RFC7575] apply. According to RFC 7575, an Autonomic Service Agent is "An agent implemented on an autonomic node that implements an autonomic function, either in part (in the case of a distributed function) or whole."

ASAs must be distinguished from other forms of software component. They are components of network or service management; they do not in themselves provide services. For example, the services envisaged for network function virtualisation [RFC8568] or for service function chaining [RFC7665] might be managed by an ASA rather than by traditional configuration tools.

The reference model [I-D.ietf-anima-reference-model] expands this by adding that an ASA is "a process that makes use of the features provided by the ANI to achieve its own goals, usually including interaction with other ASAs via the GRASP protocol [I-D.ietf-anima-grasp] or otherwise. Of course it also interacts with the specific targets of its function, using any suitable mechanism. Unless its function is very simple, the ASA will need to handle overlapping asynchronous operations. This will require either a multi-threaded implementation, or a logically equivalent event loop structure. It may therefore be a quite complex piece of software in its own right, forming part of the application layer above the ANI."

There will certainly be very simple ASAs that manage a single objective in a straightforward way and do not need asynchronous operations. In such a case, many aspects of the current document do not apply. However, in general a basic property of an ASA is that it is a relatively complex software component that will in many cases control and monitor simpler entities in the same host or elsewhere. For example, a device controller that manages tens or hundreds of simple devices might contain a single ASA.

The remainder of this document offers guidance on the design of such ASAs.

2. Logical Structure of an Autonomic Service Agent

As mentioned above, all but the simplest ASAs will need to support asynchronous operations. Not all programming environments explicitly support multi-threading. In that case, an 'event loop' style of implementation should be adopted, in which case each thread would be implemented as an event handler called in turn by the main loop. For this, the GRASP API (Section 3.3) must provide non-blocking calls. If necessary, the GRASP session identifier will be used to distinguish simultaneous operations.

A typical ASA will have a main thread that performs various initial housekeeping actions such as:

- * Obtain authorization credentials.
- * Register the ASA with GRASP.
- * Acquire relevant policy parameters.
- * Define data structures for relevant GRASP objectives.
- * Register with GRASP those objectives that it will actively manage.
- * Launch a self-monitoring thread.
- * Enter its main loop.

The logic of the main loop will depend on the details of the autonomic function concerned. Whenever asynchronous operations are required, extra threads will be launched, or events added to the event loop. Examples include:

- * Repeatedly flood an objective to the AN, so that any ASA can receive the objective's latest value.
- * Accept incoming synchronization requests for an objective managed by this ASA.
- * Accept incoming negotiation requests for an objective managed by this ASA, and then conduct the resulting negotiation with the counterpart ASA.
- * Manage subsidiary non-autonomic devices directly.

These threads or events should all either exit after their job is done, or enter a wait state for new work, to avoid blocking others unnecessarily.

According to the degree of parallelism needed by the application, some of these threads or events might be launched in multiple instances. In particular, if negotiation sessions with other ASAs are expected to be long or to involve wait states, the ASA designer might allow for multiple simultaneous negotiating threads, with appropriate use of queues and locks to maintain consistency.

The main loop itself could act as the initiator of synchronization requests or negotiation requests, when the ASA needs data or resources from other ASAs. In particular, the main loop should watch for changes in policy parameters that affect its operation. It should also do whatever is required to avoid unnecessary resource consumption, such as including an arbitrary wait time in each cycle of the main loop.

The self-monitoring thread is of considerable importance. Autonomic service agents must never fail. To a large extent this depends on careful coding and testing, with no unhandled error returns or exceptions, but if there is nevertheless some sort of failure, the self-monitoring thread should detect it, fix it if possible, and in the worst case restart the entire ASA.

Appendix B presents some example logic flows in informal pseudocode.

3. Interaction with the Autonomic Networking Infrastructure

3.1. Interaction with the security mechanisms

An ASA by definition runs in an autonomic node. Before any normal ASAs are started, such nodes must be bootstrapped into the autonomic network's secure key infrastructure in accordance with [I-D.ietf-anima-bootstrapping-keyinfra]. This key infrastructure will be used to secure the ACP (next section) and may be used by ASAs to set up additional secure interactions with their peers, if needed.

Note that the secure bootstrap process itself may include special-purpose ASAs that run in a constrained insecure mode.

3.2. Interaction with the Autonomic Control Plane

In a normal autonomic network, ASAs will run as users of the ACP, which will provide a fully secured network environment for all communication with other ASAs, in most cases mediated by GRASP (next section).

Note that the ACP formation process itself may include special-purpose ASAs that run in a constrained insecure mode.

3.3. Interaction with GRASP and its API

GRASP [I-D.ietf-anima-grasp] is expected to run as a separate process with its API [I-D.ietf-anima-grasp-api] available in user space. Thus ASAs may operate without special privilege, unless they need it for other reasons. The ASA's view of GRASP is built around GRASP objectives (Section 5), defined as data structures containing administrative information such as the objective's unique name, and its current value. The format and size of the value is not restricted by the protocol, except that it must be possible to serialise it for transmission in CBOR [RFC7049], which is no restriction at all in practice.

The GRASP API should offer the following features:

- * Registration functions, so that an ASA can register itself and the objectives that it manages.
- * A discovery function, by which an ASA can discover other ASAs supporting a given objective.
- * A negotiation request function, by which an ASA can start negotiation of an objective with a counterpart ASA. With this, there is a corresponding listening function for an ASA that wishes to respond to negotiation requests, and a set of functions to support negotiating steps.
- * A synchronization function, by which an ASA can request the current value of an objective from a counterpart ASA. With this, there is a corresponding listening function for an ASA that wishes to respond to synchronization requests.
- * A flood function, by which an ASA can cause the current value of an objective to be flooded throughout the AN so that any ASA can receive it.

For further details and some additional housekeeping functions, see [I-D.ietf-anima-grasp-api].

This API is intended to support the various interactions expected between most ASAs, such as the interactions outlined in Section 2. However, if ASAs require additional communication between themselves, they can do so using any desired protocol. One option is to use GRASP discovery and synchronization as a rendez-vous mechanism between two ASAs, passing communication parameters such as a TCP port number via GRASP. As noted above, either the ACP or in special cases the autonomic key infrastructure will be used to secure such communications.

3.4. Interaction with policy mechanism

At the time of writing, the policy (or "Intent") mechanism for the ANI is undefined and is regarded as a research topic. It is expected to operate by an information distribution mechanism (e.g. [I-D.liu-anima-grasp-distribution]) that can reach all autonomic nodes, and therefore every ASA. However, each ASA must be capable of operating "out of the box" in the absence of locally defined policy, so every ASA implementation must include carefully chosen default values and settings for all policy parameters.

4. Interaction with Non-Autonomic Components

An ASA, to have any external effects, must also interact with non-autonomic components of the node where it is installed. For example, an ASA whose purpose is to manage a resource must interact with that resource. An ASA whose purpose is to manage an entity that is already managed by local software must interact with that software. For example, if such management is performed by NETCONF [RFC6241], the ASA must interact directly with the NETCONF server in the same node. This is stating the obvious, and the details are specific to each case, but it has an important security implication. The ASA might act as a loophole by which the managed entity could penetrate the security boundary of the ANI. The ASA must be designed to avoid such loopholes, and should if possible operate in an unprivileged mode.

In an environment where systems are virtualized and specialized using techniques such as network function virtualization or network slicing, there will be a design choice whether ASAs are deployed once per physical node or once per virtual context. A related issue is whether the ANI as a whole is deployed once on a physical network, or whether several virtual ANIs are deployed. This aspect needs to be considered by the ASA designer.

5. Design of GRASP Objectives

The general rules for the format of GRASP Objective options, their names, and IANA registration are given in [I-D.ietf-anima-grasp]. Additionally that document discusses various general considerations for the design of objectives, which are not repeated here. However, we emphasize that the GRASP protocol does not provide transactional integrity. In other words, if an ASA is capable of overlapping several negotiations for a given objective, then the ASA itself must use suitable locking techniques to avoid interference between these negotiations. For example, if an ASA is allocating part of a shared resource to other ASAs, it needs to ensure that the same part of the resource is not allocated twice. This might impact the design of the objective as well as the logic flow of the ASA.

In particular, if 'dry run' mode is defined for the objective, its specification, and every implementation, must consider what state needs to be saved following a dry run negotiation, such that a subsequent live negotiation can be expected to succeed. It must be clear how long this state is kept, and what happens if the live negotiation occurs after this state is deleted. An ASA that requests a dry run negotiation must take account of the possibility that a successful dry run is followed by a failed live negotiation. Because of these complexities, the dry run mechanism should only be supported by objectives and ASAs where there is a significant benefit from it.

The actual value field of an objective is limited by the GRASP protocol definition to any data structure that can be expressed in Concise Binary Object Representation (CBOR) [RFC7049]. For some objectives, a single data item will suffice; for example an integer, a floating point number or a UTF-8 string. For more complex cases, a simple tuple structure such as [item1, item2, item3] could be used. Nothing prevents using other formats such as JSON, but this requires the ASA to be capable of parsing and generating JSON. The formats acceptable by the GRASP API will limit the options in practice. A fallback solution is for the API to accept and deliver the value field in raw CBOR, with the ASA itself encoding and decoding it via a CBOR library.

Note that a mapping from YANG to CBOR is defined by [I-D.ietf-core-yang-cbor]. Subject to the size limit defined for GRASP messages, nothing prevents objectives using YANG in this way.

6. Life Cycle

Autonomic functions could be permanent, in the sense that ASAs are shipped as part of a product and persist throughout the product's life. However, a more likely situation is that ASAs need to be installed or updated dynamically, because of new requirements or bugs. Because continuity of service is fundamental to autonomic networking, the process of seamlessly replacing a running instance of an ASA with a new version needs to be part of the ASA's design.

The implication of service continuity on the design of ASAs can be illustrated along the three main phases of the ASA life-cycle, namely Installation, Instantiation and Operation.

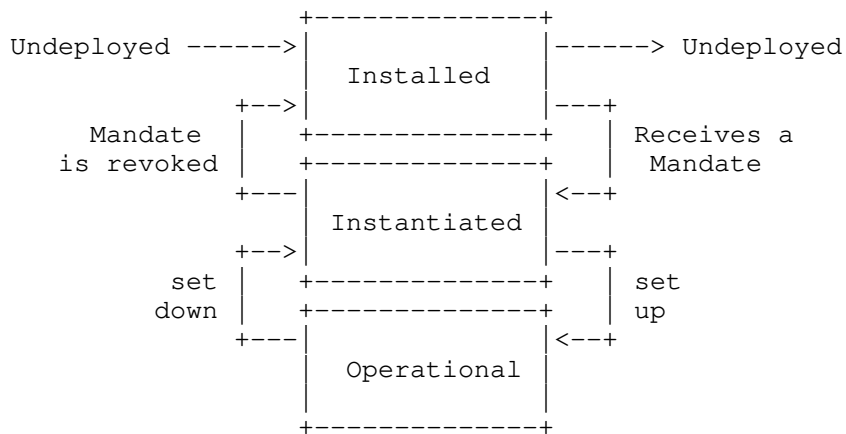


Figure 1: Life cycle of an Autonomic Service Agent

6.1. Installation phase

Before being able to instantiate and run ASAs, the operator must first provision the infrastructure with the sets of ASA software corresponding to its needs and objectives. The provisioning of the infrastructure is realized in the installation phase and consists in installing (or checking the availability of) the pieces of software of the different ASA classes in a set of Installation Hosts.

There are 3 properties applicable to the installation of ASAs:

The dynamic installation property allows installing an ASA on demand, on any hosts compatible with the ASA.

The decoupling property allows controlling resources of a NE from a remote ASA, i.e. an ASA installed on a host machine different from the resources' NE.

The multiplicity property allows controlling multiple sets of resources from a single ASA.

These three properties are very important in the context of the installation phase as their variations condition how the ASA class could be installed on the infrastructure.

6.1.1. Installation phase inputs and outputs

Inputs are:

[ASA class of type_x] that specifies which classes ASAs to install,

[Installation_target_Infrastructure] that specifies the candidate Installation Hosts,

[ASA class placement function, e.g. under which criteria/ constraints as defined by the operator] that specifies how the installation phase shall meet the operator's needs and objectives for the provision of the infrastructure. In the coupled mode, the placement function is not necessary, whereas in the decoupled mode, the placement function is mandatory, even though it can be as simple as an explicit list of Installation hosts.

The main output of the installation phase is an up-to-date directory of installed ASAs which corresponds to [list of ASA classes] installed on [list of installation Hosts]. This output is also useful for the coordination function and corresponds to the static interaction map (see next section).

The condition to validate in order to pass to next phase is to ensure that [list of ASA classes] are well installed on [list of installation Hosts]. The state of the ASA at the end of the installation phase is: installed. (not instantiated). The following commands or messages are foreseen: install(list of ASA classes, Installation_target_Infrastructure, ASA class placement function), and un-install (list of ASA classes).

6.2. Instantiation phase

Once the ASAs are installed on the appropriate hosts in the network, these ASA may start to operate. From the operator viewpoint, an operating ASA means the ASA manages the network resources as per the objectives given. At the ASA local level, operating means executing their control loop/algorithm.

But right before that, there are two things to take into consideration. First, there is a difference between 1. having a piece of code available to run on a host and 2. having an agent based on this piece of code running inside the host. Second, in a coupled case, determining which resources are controlled by an ASA is straightforward (the determination is embedded), in a decoupled mode determining this is a bit more complex (hence a starting agent will have to either discover or be taught it).

The instantiation phase of an ASA covers both these aspects: starting the agent piece of code (when this does not start automatically) and determining which resources have to be controlled (when this is not obvious).

6.2.1. Operator's goal

Through this phase, the operator wants to control its autonomic network in two things:

- 1 determine the scope of autonomic functions by instructing which of the network resources have to be managed by which autonomic function (and more precisely which class e.g. 1. version X or version Y or 2. provider A or provider B),
- 2 determine how the autonomic functions are organized by instructing which ASAs have to interact with which other ASAs (or more precisely which set of network resources have to be handled as an autonomous group by their managing ASAs).

Additionally in this phase, the operator may want to set objectives to autonomic functions, by configuring the ASAs technical objectives.

The operator's goal can be summarized in an instruction to the ANIMA ecosystem matching the following pattern:

```
[ASA of type_x instances] ready to control  
[Instantiation_target_Infrastructure] with  
[Instantiation_target_parameters]
```

6.2.2. Instantiation phase inputs and outputs

Inputs are:

[ASA of type_x instances] that specifies which are the ASAs to be targeted (and more precisely which class e.g. 1. version X or version Y or 2. provider A or provider B),

[Instantiation_target_Infrastructure] that specifies which are the resources to be managed by the autonomic function, this can be the whole network or a subset of it like a domain a technology segment or even a specific list of resources,

[Instantiation_target_parameters] that specifies which are the technical objectives to be set to ASAs (e.g. an optimization target)

Outputs are:

[Set of ASAs - Resources relations] describing which resources are managed by which ASA instances, this is not a formal message, but a resulting configuration of a set of ASAs,

6.2.3. Instantiation phase requirements

The instructions described in section 4.2 could be either:

sent to a targeted ASA In which case, the receiving Agent will have to manage the specified list of
[Instantiation_target_Infrastructure], with the
[Instantiation_target_parameters].

broadcast to all ASAs In which case, the ASAs would collectively determine from the list which Agent(s) would handle which
[Instantiation_target_Infrastructure], with the
[Instantiation_target_parameters].

This set of instructions can be materialized through a message that is named an Instance Mandate (description TBD).

The conclusion of this instantiation phase is a ready to operate ASA (or interacting set of ASAs), then this (or those) ASA(s) can describe themselves by depicting which are the resources they manage and what this means in terms of metrics being monitored and in terms of actions that can be executed (like modifying the parameters values). A message conveying such a self description is named an Instance Manifest (description TBD).

Though the operator may well use such a self-description "per se", the final goal of such a description is to be shared with other ANIMA entities like:

- * the coordination entities (see [I-D.ciavaglia-anima-coordination])
- * collaborative entities in the purpose of establishing knowledge exchanges (some ASAs may produce knowledge or even monitor metrics that other ASAs cannot make by themselves why those would be useful for their execution)

6.3. Operation phase

Note: This section is to be further developed in future revisions of the document, especially the implications on the design of ASAs.

During the Operation phase, the operator can:

Activate/Deactivate ASA: meaning enabling those to execute their autonomic loop or not.

Modify ASAs targets: meaning setting them different objectives.

Modify ASAs managed resources: by updating the instance mandate which would specify different set of resources to manage (only applicable to decouples ASAs).

During the Operation phase, running ASAs can interact the one with the other:

in order to exchange knowledge (e.g. an ASA providing traffic predictions to load balancing ASA)

in order to collaboratively reach an objective (e.g. ASAs pertaining to the same autonomic function targeted to manage a network domain, these ASA will collaborate - in the case of a load balancing one, by modifying the links metrics according to the neighboring resources loads)

During the Operation phase, running ASAs are expected to apply coordination schemes

then execute their control loop under coordination supervision/instructions

The ASA life-cycle is discussed in more detail in "A Day in the Life of an Autonomic Function" [I-D.peloso-anima-autonomic-function].

7. Coordination between Autonomic Functions

Some autonomic functions will be completely independent of each other. However, others are at risk of interfering with each other - for example, two different optimization functions might both attempt to modify the same underlying parameter in different ways. In a complete system, a method is needed of identifying ASAs that might interfere with each other and coordinating their actions when necessary. This issue is considered in "Autonomic Functions Coordination" [I-D.ciavaglia-anima-coordination].

8. Coordination with Traditional Management Functions

Some ASAs will have functions that overlap with existing configuration tools and network management mechanisms such as command line interfaces, DHCP, DHCPv6, SNMP, NETCONF, RESTCONF and YANG-based solutions. Each ASA designer will need to consider this issue and how to avoid clashes and inconsistencies. Some specific considerations for interaction with OAM tools are given in [RFC8368]. As another example, [I-D.ietf-anima-prefix-management] describes how autonomic management of IPv6 prefixes can interact with prefix delegation via DHCPv6. The description of a GRASP objective and of an ASA using it should include a discussion of any such interactions.

A related aspect is that management functions often include a data model, quite likely to be expressed in a formal notation such as YANG. This aspect should not be an afterthought in the design of an ASA. To the contrary, the design of the ASA and of its GRASP objectives should match the data model; as noted above, YANG serialized as CBOR may be used directly as the value of a GRASP objective.

9. Robustness

It is of great importance that all components of an autonomic system are highly robust. In principle they must never fail. This section lists various aspects of robustness that ASA designers should consider.

1. If despite all precautions, an ASA does encounter a fatal error, it should in any case restart automatically and try again. To mitigate a hard loop in case of persistent failure, a suitable pause should be inserted before such a restart. The length of the pause depends on the use case.
2. If a newly received or calculated value for a parameter falls out of bounds, the corresponding parameter should be either left unchanged or restored to a safe value.

3. If a GRASP synchronization or negotiation session fails for any reason, it may be repeated after a suitable pause. The length of the pause depends on the use case.
4. If a session fails repeatedly, the ASA should consider that its peer has failed, and cause GRASP to flush its discovery cache and repeat peer discovery.
5. In any case, it may be prudent to repeat discovery periodically, depending on the use case.
6. Any received GRASP message should be checked. If it is wrongly formatted, it should be ignored. Within a unicast session, an Invalid message (M_INVALID) may be sent. This function may be provided by the GRASP implementation itself.
7. Any received GRASP objective should be checked. If it is wrongly formatted, it should be ignored. Within a negotiation session, a Negotiation End message (M_END) with a Decline option (O_DECLINE) should be sent. An ASA may log such events for diagnostic purposes.
8. If an ASA receives either an Invalid message (M_INVALID) or a Negotiation End message (M_END) with a Decline option (O_DECLINE), one possible reason is that the peer ASA does not support a new feature of either GRASP or of the objective in question. In such a case the ASA may choose to repeat the operation concerned without using that new feature.
9. All other possible exceptions should be handled in an orderly way. There should be no such thing as an unhandled exception (but see point 1 above).
10. Security Considerations

ASAs are intended to run in an environment that is protected by the Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane], admission to which depends on an initial secure bootstrap process [I-D.ietf-anima-bootstrapping-keyinfra]. In some deployments, a secure partition of the link layer might be used instead [I-D.carpenter-anima-l2acp-scenarios]. However, this does not relieve ASAs of responsibility for security. In particular, when ASAs configure or manage network elements outside the ACP, they must use secure techniques and carefully validate any incoming information. As noted above, this will apply in particular when an ASA interacts with a management component such as a NETCONF server.

As appropriate to their specific functions, ASAs should take account of relevant privacy considerations [RFC6973].

Authorization of ASAs is a subject for future study. At present, ASAs are trusted by virtue of being installed on a node that has successfully joined the ACP. In the general case, a node may have multiple roles and a role may use multiple ASAs, each using multiple GRASP objectives. Additional mechanisms for the authorization of nodes and ASAs to manipulate specific GRASP objectives could be designed.

11. IANA Considerations

This document makes no request of the IANA.

12. Acknowledgements

Useful comments were received from Michael Behringer Toerless Eckert, Alex Galis, Bing Liu, Michael Richardson, and other members of the ANIMA WG.

13. References

13.1. Normative References

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

Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-27, 2 July 2020, <<https://tools.ietf.org/html/draft-ietf-anima-autonomic-control-plane-27>>.

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

Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-bootstrapping-keyinfra-41, 8 April 2020, <<https://tools.ietf.org/html/draft-ietf-anima-bootstrapping-keyinfra-41>>.

[I-D.ietf-anima-grasp]

Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017, <<https://tools.ietf.org/html/draft-ietf-anima-grasp-15>>.

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

13.2. Informative References

- [DeMola06] De Mola, F. and R. Quitadamo, "An Agent Model for Future Autonomic Communications", Proceedings of the 7th WOA 2006 Workshop From Objects to Agents 51-59, September 2006.
- [GANA13] "Autonomic network engineering for the self-managing Future Internet (AFI): GANA Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management.", April 2013, <http://www.etsi.org/deliver/etsi_gs/AFI/001_099/002/01.01.01_60/gs_afi002v010101p.pdf>.
- [Huebscher08] Huebscher, M. C. and J. A. McCann, "A survey of autonomic computing--degrees, models, and applications", ACM Computing Surveys (CSUR) Volume 40 Issue 3 DOI: 10.1145/1380584.1380585, August 2008.
- [I-D.carpenter-anima-l2acp-scenarios] Carpenter, B. and B. Liu, "Scenarios and Requirements for Layer 2 Autonomic Control Planes", Work in Progress, Internet-Draft, draft-carpenter-anima-l2acp-scenarios-02, 8 April 2020, <<https://tools.ietf.org/html/draft-carpenter-anima-l2acp-scenarios-02>>.
- [I-D.ciavaglia-anima-coordination] Ciavaglia, L. and P. Peloso, "Autonomic Functions Coordination", Work in Progress, Internet-Draft, draft-ciavaglia-anima-coordination-01, 21 March 2016, <<https://tools.ietf.org/html/draft-ciavaglia-anima-coordination-01>>.
- [I-D.ietf-anima-grasp-api] Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-api-06, 12 June 2020, <<https://tools.ietf.org/html/draft-ietf-anima-grasp-api-06>>.
- [I-D.ietf-anima-prefix-management] Jiang, S., Du, Z., Carpenter, B., and Q. Sun, "Autonomic IPv6 Edge Prefix Management in Large-scale Networks", Work

in Progress, Internet-Draft, draft-ietf-anima-prefix-management-07, 18 December 2017,
<<https://tools.ietf.org/html/draft-ietf-anima-prefix-management-07>>.

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

Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", Work in Progress, Internet-Draft, draft-ietf-anima-reference-model-10, 22 November 2018,
<<https://tools.ietf.org/html/draft-ietf-anima-reference-model-10>>.

[I-D.ietf-core-yang-cbor]

Veillette, M., Petrov, I., and A. Pelov, "CBOR Encoding of Data Modeled with YANG", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-13, 4 July 2020,
<<https://tools.ietf.org/html/draft-ietf-core-yang-cbor-13>>.

[I-D.liu-anima-grasp-distribution]

Liu, B., Xiao, X., Hecker, A., Jiang, S., and Z. Despotovic, "Information Distribution in Autonomic Networking", Work in Progress, Internet-Draft, draft-liu-anima-grasp-distribution-13, 12 December 2019,
<<https://tools.ietf.org/html/draft-liu-anima-grasp-distribution-13>>.

[I-D.peloso-anima-autonomic-function]

Pierre, P. and L. Ciavaglia, "A Day in the Life of an Autonomic Function", Work in Progress, Internet-Draft, draft-peloso-anima-autonomic-function-01, 21 March 2016,
<<https://tools.ietf.org/html/draft-peloso-anima-autonomic-function-01>>.

[Movahedi12]

Movahedi, Z., Ayari, M., Langar, R., and G. Pujolle, "A Survey of Autonomic Network Architectures and Evaluation Criteria", IEEE Communications Surveys & Tutorials Volume: 14 , Issue: 2 DOI: 10.1109/SURV.2011.042711.00078, Page(s): 464 - 490, 2012.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
<<https://www.rfc-editor.org/info/rfc6241>>.

- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.
- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <<https://www.rfc-editor.org/info/rfc8368>>.
- [RFC8568] Bernardos, CJ., Rahman, A., Zuniga, JC., Contreras, LM., Aranda, P., and P. Lynch, "Network Virtualization Research Challenges", RFC 8568, DOI 10.17487/RFC8568, April 2019, <<https://www.rfc-editor.org/info/rfc8568>>.

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

draft-carpenter-anima-asa-guidelines-09, 2020-07-25:

- * Additional text on future authorization.
- * Editorial fixes

draft-carpenter-anima-asa-guidelines-08, 2020-01-10:

- * Introduced notion of autonomic ecosystem.
- * Minor technical clarifications.
- * Converted to v3 format.

draft-carpenter-anima-asa-guidelines-07, 2019-07-17:

- * Improved explanation of threading vs event-loop
- * Other editorial improvements.

draft-carpenter-anima-asa-guidelines-06, 2018-01-07:

- * Expanded and improved example logic flow.
- * Editorial corrections.

draft-carpenter-anima-asa-guidelines-05, 2018-06-30:

- * Added section on relationship with non-autonomic components.
- * Editorial corrections.

draft-carpenter-anima-asa-guidelines-04, 2018-03-03:

- * Added note about simple ASAs.
- * Added note about NFV/SFC services.
- * Improved text about threading v event loop model
- * Added section about coordination with traditional tools.
- * Added appendix with example logic flow.

draft-carpenter-anima-asa-guidelines-03, 2017-10-25:

- * Added details on life cycle.
- * Added details on robustness.
- * Added co-authors.

draft-carpenter-anima-asa-guidelines-02, 2017-07-01:

- * Expanded description of event-loop case.
- * Added note about 'dry run' mode.

draft-carpenter-anima-asa-guidelines-01, 2017-01-06:

- * More sections filled in.

draft-carpenter-anima-asa-guidelines-00, 2016-09-30:

- * Initial version

Appendix B. Example Logic Flows

This appendix describes generic logic flows for an Autonomic Service Agent (ASA) for resource management. Note that these are illustrative examples, and in no sense requirements. As long as the rules of GRASP are followed, a real implementation could be different. The reader is assumed to be familiar with GRASP [I-D.ietf-anima-grasp] and its conceptual API [I-D.ietf-anima-grasp-api].

A complete autonomic function for a resource would consist of a number of instances of the ASA placed at relevant points in a network. Specific details will of course depend on the resource

concerned. One example is IP address prefix management, as specified in [I-D.ietf-anima-prefix-management]. In this case, an instance of the ASA would exist in each delegating router.

An underlying assumption is that there is an initial source of the resource in question, referred to here as an origin ASA. The other ASAs, known as delegators, obtain supplies of the resource from the origin, and then delegate quantities of the resource to consumers that request it, and recover it when no longer needed.

Another assumption is there is a set of network wide policy parameters, which the origin will provide to the delegators. These parameters will control how the delegators decide how much resource to provide to consumers. Thus the ASA logic has two operating modes: origin and delegator. When running as an origin, it starts by obtaining a quantity of the resource from the NOC, and it acts as a source of policy parameters, via both GRASP flooding and GRASP synchronization. (In some scenarios, flooding or synchronization alone might be sufficient, but this example includes both.)

When running as a delegator, it starts with an empty resource pool, it acquires the policy parameters by GRASP synchronization, and it delegates quantities of the resource to consumers that request it. Both as an origin and as a delegator, when its pool is low it seeks quantities of the resource by requesting GRASP negotiation with peer ASAs. When its pool is sufficient, it hands out resource to peer ASAs in response to negotiation requests. Thus, over time, the initial resource pool held by the origin will be shared among all the delegators according to demand.

In theory a network could include any number of origins and any number of delegators, with the only condition being that each origin's initial resource pool is unique. A realistic scenario is to have exactly one origin and as many delegators as you like. A scenario with no origin is useless.

An implementation requirement is that resource pools are kept in stable storage. Otherwise, if a delegator exits for any reason, all the resources it has obtained or delegated are lost. If an origin exits, its entire spare pool is lost. The logic for using stable storage and for crash recovery is not included in the pseudocode below.

The description below does not implement GRASP's 'dry run' function. That would require temporarily marking any resource handed out in a dry run negotiation as reserved, until either the peer obtains it in a live run, or a suitable timeout expires.

The main data structures used in each instance of the ASA are:

- * The `resource_pool`, for example an ordered list of available resources. Depending on the nature of the resource, units of resource are split when appropriate, and a background garbage collector recombines split resources if they are returned to the pool.
- * The `delegated_list`, where a delegator stores the resources it has given to consumers routers.

Possible main logic flows are below, using a threaded implementation model. The transformation to an event loop model should be apparent – each thread would correspond to one event in the event loop.

The GRASP objectives are as follows:

- * `["EX1.Resource", flags, loop_count, value]` where the value depends on the resource concerned, but will typically include its size and identification.
- * `["EX1.Params", flags, loop_count, value]` where the value will be, for example, a JSON object defining the applicable parameters.

In the outline logic flows below, these objectives are represented simply by their names.

<CODE BEGINS>

MAIN PROGRAM:

```
Create empty resource_pool (and an associated lock)
Create empty delegated_list
Determine whether to act as origin
if origin:
    Obtain initial resource_pool contents from NOC
    Obtain value of EX1.Params from NOC
Register ASA with GRASP
Register GRASP objectives EX1.Resource and EX1.Params
if origin:
    Start FLOODER thread to flood EX1.Params
    Start SYNCHRONIZER listener for EX1.Params
Start MAIN_NEGOTIATOR thread for EX1.Resource
if not origin:
    Obtain value of EX1.Params from GRASP flood or synchronization
    Start DELEGATOR thread
Start GARBAGE_COLLECTOR thread
do forever:
    good_peer = none
    if resource_pool is low:
        Calculate amount A of resource needed
        Discover peers using GRASP M_DISCOVER / M_RESPONSE
        if good_peer in peers:
            peer = good_peer
        else:
            peer = #any choice among peers
            grasp.request_negotiate("EX1.Resource", peer)
            i.e., send M_REQ_NEG
            Wait for response (M_NEGOTIATE, M_END or M_WAIT)
            if OK:
                if offered amount of resource sufficient:
                    Send M_END + O_ACCEPT #negotiation succeeded
                    Add resource to pool
                    good_peer = peer
                else:
                    Send M_END + O_DECLINE #negotiation failed
            sleep() #sleep time depends on application scenario
```

MAIN_NEGOTIATOR thread:

```
do forever:
    grasp.listen_negotiate("EX1.Resource")
    i.e., wait for M_REQ_NEG
    Start a separate new NEGOTIATOR thread for requested amount A
```

NEGOTIATOR thread:

```
Request resource amount A from resource_pool
if not OK:
    while not OK and A > Amin:
        A = A-1
        Request resource amount A from resource_pool
if OK:
    Offer resource amount A to peer by GRASP M_NEGOTIATE
    if received M_END + O_ACCEPT:
        #negotiation succeeded
    elif received M_END + O_DECLINE or other error:
        #negotiation failed
else:
    Send M_END + O_DECLINE #negotiation failed
```

DELEGATOR thread:

```
do forever:
    Wait for request or release for resource amount A
    if request:
        Get resource amount A from resource_pool
        if OK:
            Delegate resource to consumer
            Record in delegated_list
        else:
            Signal failure to consumer
            Signal main thread that resource_pool is low
    else:
        Delete resource from delegated_list
        Return resource amount A to resource_pool
```

SYNCHRONIZER thread:

```
do forever:
    Wait for M_REQ_SYN message for EX1.Params
    Reply with M_SYNCH message for EX1.Params
```

FLOODER thread:

```
do forever:
    Send M_FLOOD message for EX1.Params
    sleep() #sleep time depends on application scenario
```

GARBAGE_COLLECTOR thread:

```
do forever:
    Search resource_pool for adjacent resources
    Merge adjacent resources
    sleep() #sleep time depends on application scenario
```

<CODE ENDS>

Authors' Addresses

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

Email: brian.e.carpenter@gmail.com

Laurent Ciavaglia
Nokia
Villardeaux
91460 Nozay
France

Email: laurent.ciavaglia@nokia.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14 Huawei Campus
156 Beiqing Road
Hai-Dian District
Beijing
100095
China

Email: jiangsheng@huawei.com

Pierre Peloso
Nokia
Villardeaux
91460 Nozay
France

Email: pierre.peloso@nokia.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 11 October 2020

B. E. Carpenter
Univ. of Auckland
B. Liu
Huawei Technologies
9 April 2020

Scenarios and Requirements for Layer 2 Autonomic Control Planes
draft-carpenter-anima-l2acp-scenarios-02

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 an autonomic network and for the Generic Autonomic Signaling Protocol (GRASP).

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 11 October 2020.

Copyright Notice

Copyright (c) 2020 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	2
2. Network Scenarios Suitable for a Layer 2 ACP	3
3. Requirements for a Layer 2 Technology	3
4. Multiple Segments	5
5. Implementation Status [RFC Editor: please remove]	5
6. Security Considerations	5
7. IANA Considerations	5
8. Acknowledgements	5
9. References	5
9.1. Normative References	5
9.2. Informative References	6
Appendix A. Change log [RFC Editor: Please remove]	7
Authors' Addresses	7

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 Network Infrastructure (ANI).

For this reason, GRASP, and any other autonomic management traffic, 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. It is not intended to be a normative specification, since implementation details will depend on individual layer 2 technologies.

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, which 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, which also requires 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

These requirements are intended to ensure that a layer 2 ACP can meet the needs of all components of the ANI.

1. Since GRASP is specified to run over IPv6, 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. Note that since GRASP is specified to run unicast operations over TCP, this is not an absolute requirement and the IPv6 minimum MTU of 1280 bytes would be acceptable. GRASP UDP multicast messages could in principle be fragmented but in normal operation this would be unusual.

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 with 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 specific security protocol that supports both authentication and encryption of layer 2 packets for Ethernet LANs is MACsec, i.e. the IEEE Standard 802.1AE-2018 [MACsec]. For multicast packets, authentication is on a group basis (i.e., the originator is guaranteed to be a member of the group, rather than a specific interface). MACsec applies across all VLANs, but the ACP VLAN can be isolated from the data plane VLAN independently of MACsec. This solution does not extend to wireless networks. For IEEE 802.11 networks, IEEE Standard 802.11-2016 [WiFi] "WPA2" security within a dedicated Basic Service Set (BSS) might be considered adequate.

An ACP software module will be needed in each autonomic node, whose job is to provide the GRASP core or other autonomic management protocols 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 (also known as a zone index [RFC4007]) is an integer for maximum portability between operating systems.

4. Multiple Segments

The L2 ACP could in principle be extended across multiple segments or even multiple sites by use of secure L2VPN technology. This topic is out of the scope of the present document.

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 Michael Richardson and other participants in the ANIMA WG.

9. References

9.1. Normative References

- [MACsec] "IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Security", IEEE Standard 802.1AE-2018, 2018, <<https://ieeexplore.ieee.org/browse/standards/get-program/page/series?id=68>>.

- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <<https://www.rfc-editor.org/info/rfc4007>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [WiFi] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11-2016, 2016, <<http://standards.ieee.org/getieee802/download/80211-2016.pdf>>.

9.2. Informative References

- [I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-24, 9 March 2020, <<https://tools.ietf.org/html/draft-ietf-anima-autonomic-control-plane-24>>.
- [I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017, <<https://tools.ietf.org/html/draft-ietf-anima-grasp-15>>.
- [I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", Work in Progress, Internet-Draft, draft-ietf-anima-reference-model-10, 22 November 2018, <<https://tools.ietf.org/html/draft-ietf-anima-reference-model-10>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network

Operations, Administration, and Maintenance (OAM)",
RFC 8368, DOI 10.17487/RFC8368, May 2018,
<<https://www.rfc-editor.org/info/rfc8368>>.

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

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

- * Initial version

draft-carpenter-anima-l2acp-scenarios-01, 2019-10-03:

- * Added discussion of MACsec and WPA2
- * Editorial improvements

draft-carpenter-anima-l2acp-scenarios-02, 2020-04-09:

- * Updated references
- * Editorial improvements
- * Converted to xml2rfc v3

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
Huawei Technologies
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing
100095
P.R. China

Email: leo.liubing@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 26, 2020

O. Friel
R. Barnes
Cisco
R. Shekh-Yusef
Avaya
October 24, 2019

ACME Integrations
draft-friel-acme-integrations-02

Abstract

This document outlines multiple advanced use cases and integrations that ACME facilitates without any modifications or enhancements required to the base ACME specification. The use cases include ACME integration with EST, BRSKI and TEAP.

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 26, 2020.

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	2
2. Terminology	3
3. ACME Integration with EST	3
4. ACME Integration with BRSKI	6
5. ACME Integration with BRSKI Default Cloud Registrar	8
6. ACME Integration with TEAP	10
7. ACME Integration with TEAP-BRSKI	13
8. IANA Considerations	16
9. Security Considerations	16
10. Informative References	16
Appendix A. Comments	17
Authors' Addresses	17

1. Introduction

ACME [RFC8555] defines a protocol that a certificate authority (CA) and an applicant can use to automate the process of domain name ownership validation and X.509 (PKIX) certificate issuance. The protocol is rich and flexible and enables multiple use cases that are not immediately obvious from reading the specification. This document explicitly outlines multiple advanced ACME use cases including:

- o ACME integration with EST [RFC7030]
- o ACME integration with BRSKI
[I-D.ietf-anima-bootstrapping-keyinfra]
- o ACME integration with BRSKI Default Cloud Registrar
[I-D.friel-anima-brski-cloud]
- o ACME integration with TEAP [RFC7170]
- o ACME integration with TEAP-BRSKI [I-D.lear-eap-teap-brski]

The integrations with EST, BRSKI (which is based upon EST), and TEAP enable automated certificate enrolment for devices. ACME for subdomains [I-D.friel-acme-subdomains] outlines how ACME can be used by a client to obtain a certificate for a subdomain identifier from a certificate authority where client has fulfilled a challenge against a parent domain but does not need to fulfil a challenge against the explicit subdomain. This is a useful optimisation when ACME is used to issue certificates for large numbers of devices as it reduces the

domain ownership proof traffic (DNS or HTTP) and ACME traffic overhead, but is not a necessary requirement.

2. Terminology

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

The following terms are used in this document:

- o BRSKI: Bootstrapping Remote Secure Key Infrastructures [I-D.ietf-anima-bootstrapping-keyinfra]
- o CA: Certificate Authority
- o CMC: Certificate Management over CMS
- o CSR: Certificate Signing Request
- o EST: Enrollment over Secure Transport [RFC7030]
- o FQDN: Fully Qualified Domain Name
- o RA: PKI Registration Authority
- o TEAP: Tunneled Extensible Authentication Protocol [RFC7170]

3. ACME Integration with EST

EST [RFC7030] defines a mechanism for clients to enroll with a PKI Registration Authority by sending CMC messages over HTTP. EST section 1 states:

"Architecturally, the EST service is located between a Certification Authority (CA) and a client. It performs several functions traditionally allocated to the Registration Authority (RA) role in a PKI."

EST section 1.1 states that:

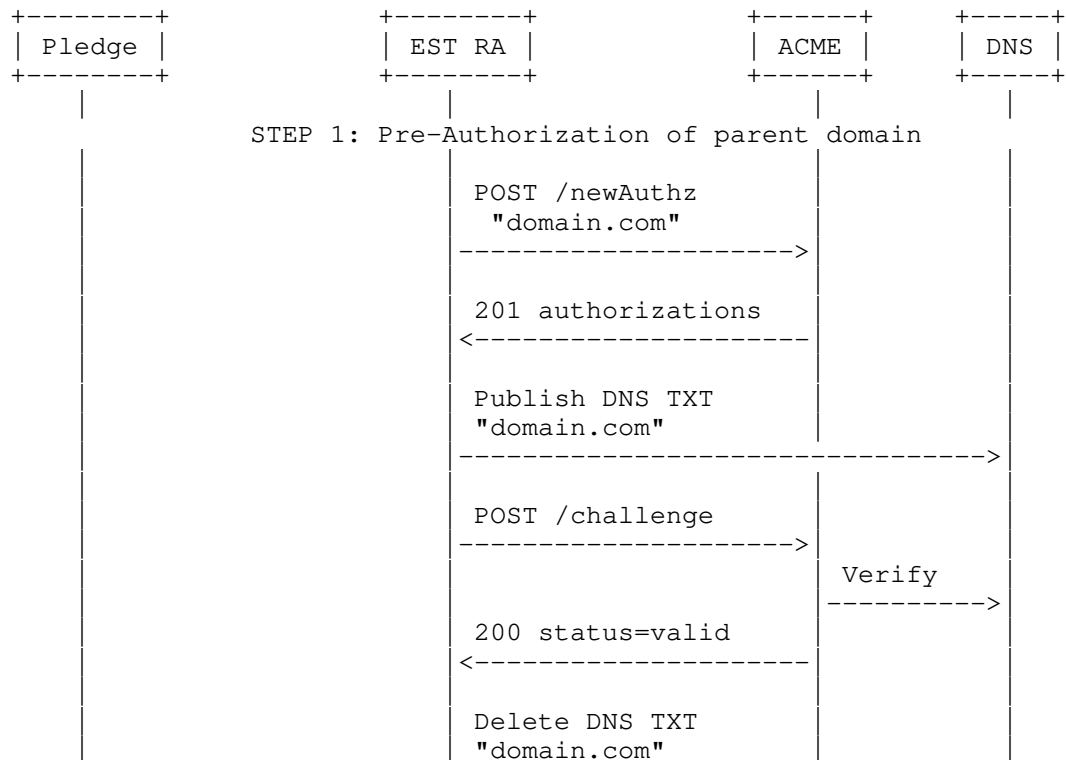
"For certificate issuing services, the EST CA is reached through the EST server; the CA could be logically "behind" the EST server or embedded within it."

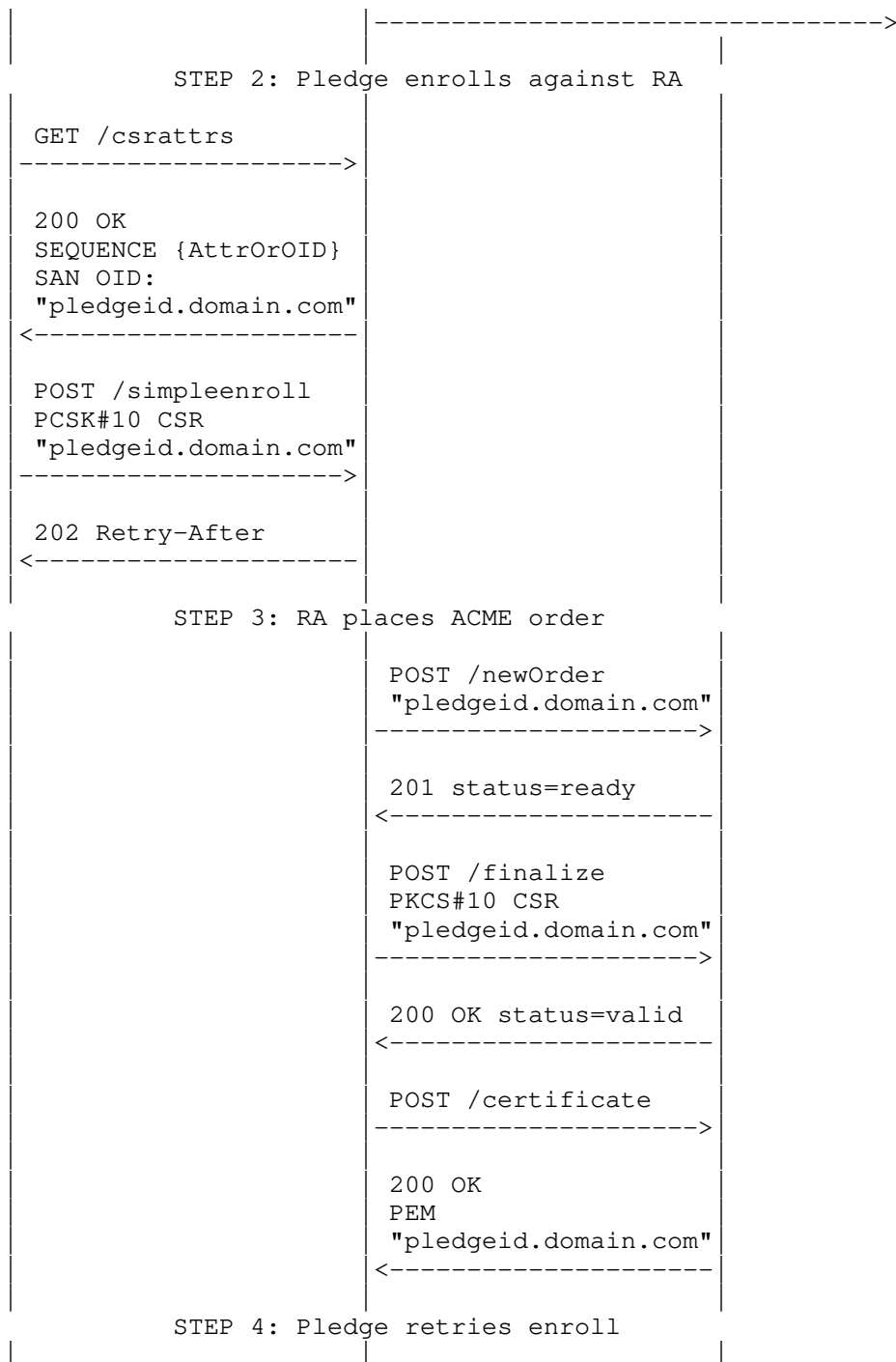
When the CA is logically "behind" the EST RA, EST does not specify how the RA communicates with the CA. EST section 1 states:

"The nature of communication between an EST server and a CA is not described in this document."

This section outlines how ACME could be used for communication between the EST RA and the CA. The example call flow leverages [I-D.friel-acme-subdomains] and shows the RA proving ownership of a parent domain, with individual client certificates being subdomains under that parent domain. This is an optimisation that reduces DNS and ACME traffic overhead. The RA could of course prove ownership of every explicit client certificate identifier.

The call flow illustrates the client calling the EST /csrattrs API before calling the EST /simpleenroll API. This enables the EST server to indicate to the client what attributes it expects the client to include in the CSR request send in the /simpleenroll API. For example, EST servers could use this mechanism to tell the client what fields to include in the CSR Subject and Subject Alternative Name fields.





```

POST /simpleenroll
PCSK#10 CSR
"pledgeid.domain.com"
----->

200 OK
PKCS#7
"pledgeid.domain.com"
<-----

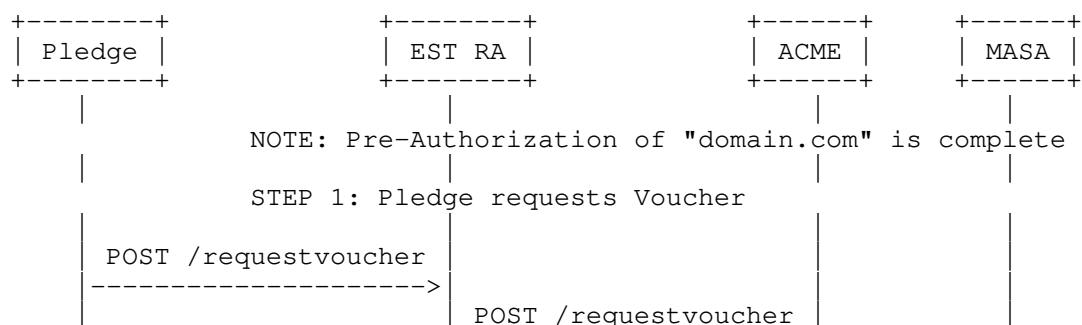
```

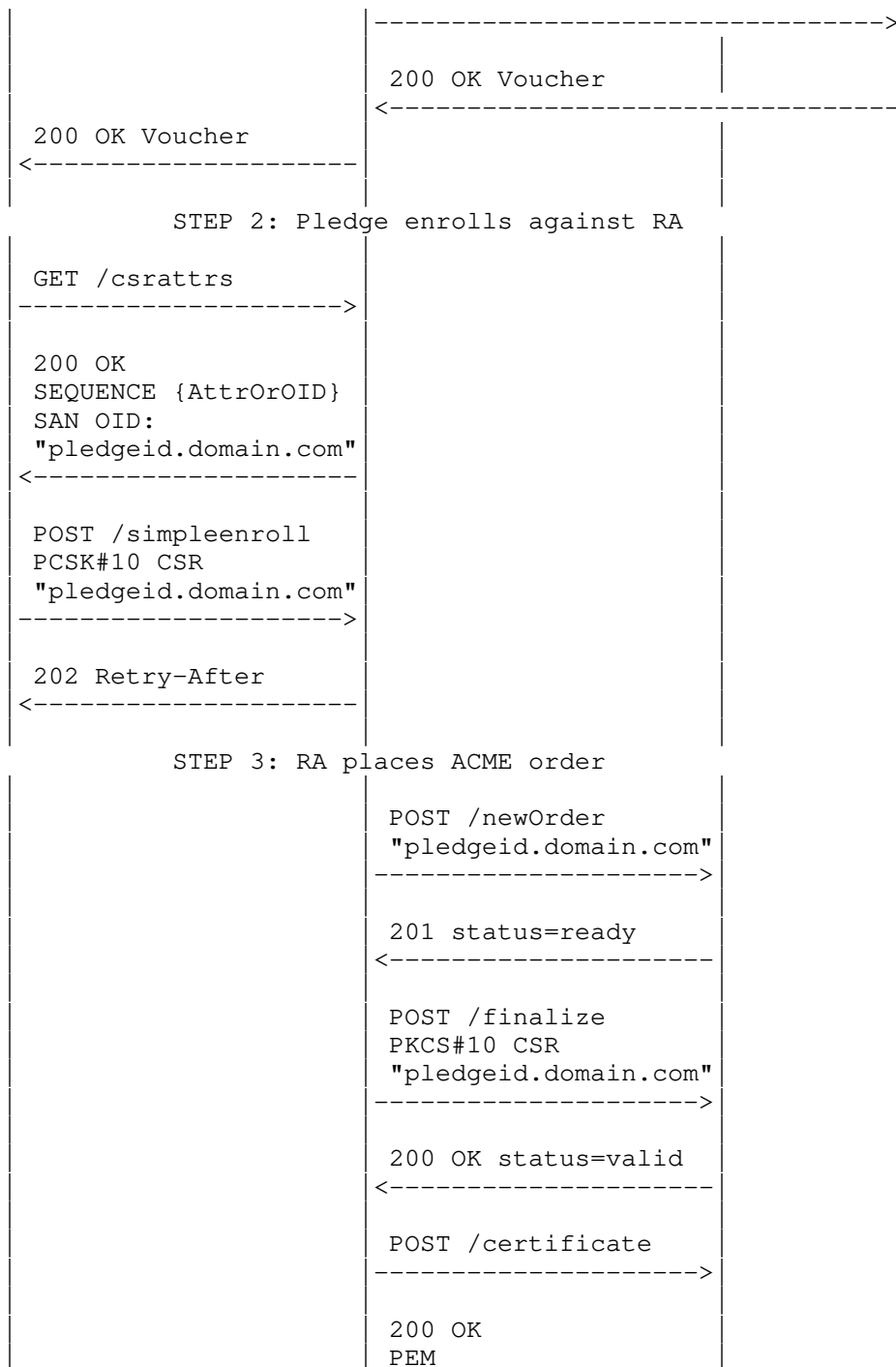
4. ACME Integration with BRSKI

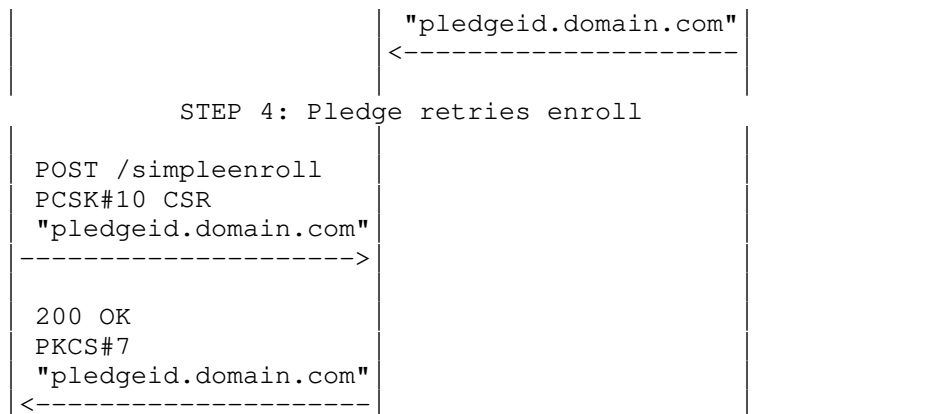
BRSKI [I-D.ietf-anima-bootstrapping-keyinfra] is based upon EST [RFC7030] and defines how to autonomically bootstrap PKI trust anchors into devices via means of signed vouchers. EST certificate enrollment may then optionally take place after trust has been established. BRSKI voucher exchange and trust establishment are based on EST extensions and the certificate enrollment part of BRSKI is fully based on EST. Similar to EST, BRSKI does not define how the EST RA communicates with the CA. Therefore, the mechanisms outlined in the previous section for using ACME as the communications protocol between the EST RA and the CA are equally applicable to BRSKI.

The following call flow shows how ACME may be integrated into a full BRSKI voucher plus EST enrollment workflow. For brevity, it assumes that the EST RA has previously proven ownership of a parent domain and that pledge certificate identifiers are a subdomain of that parent domain. The domain ownership exchanges between the RA, ACME and DNS are not shown. Similarly, not all BRSKI interactions are shown and only the key protocol flows involving voucher exchange and EST enrollment are shown.

Similar to the EST section above, the client calls EST /csrattrs API before calling the EST /simpleenroll API. This enables the server to indicate what fields the pledge should include in the CSR that the client sends in the /simpleenroll API.



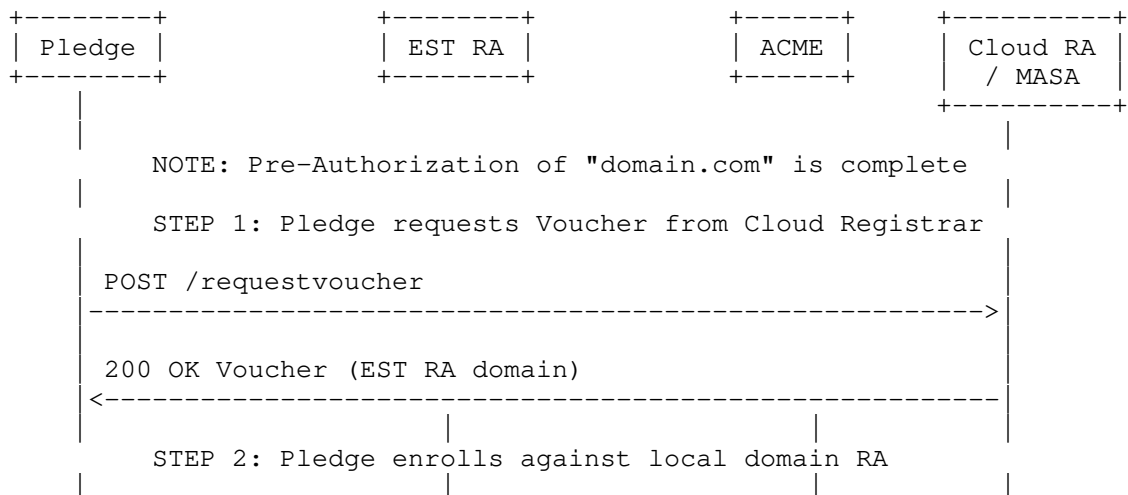


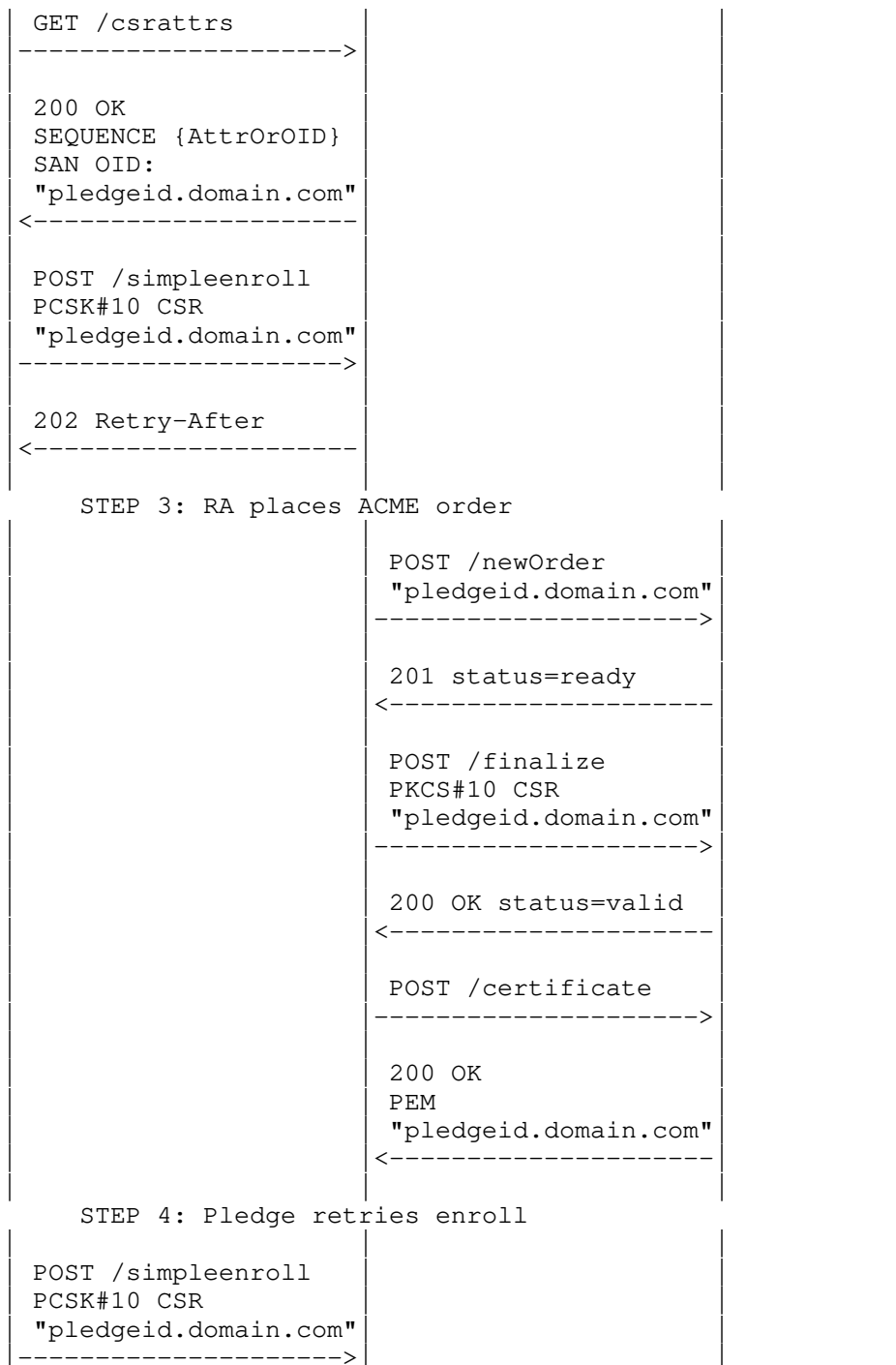


5. ACME Integration with BRSKI Default Cloud Registrar

BRSKI Cloud Registrar [I-D.friel-anima-brski-cloud] specifies the behaviour of a BRSKI Cloud Registrar, and how a pledge can interact with a BRSKI Cloud Registrar when bootstrapping. Similar to the local domain registrar BRSKI flow, ACME can be easily integrated with a cloud registrar bootstrap flow.

BRSKI cloud registrar is flexible and allows for multiple different local domain discovery and redirect scenarios. In the example illustrated here, the extension to [RFC8366] Vouchers which is defined in [[TODO ID-TBD]] and allows the specification of a bootstrap DNS domain is leveraged. This extension allows the cloud registrar to specify the local domain RA that the pledge should connect to for the purposes of EST enrollment.





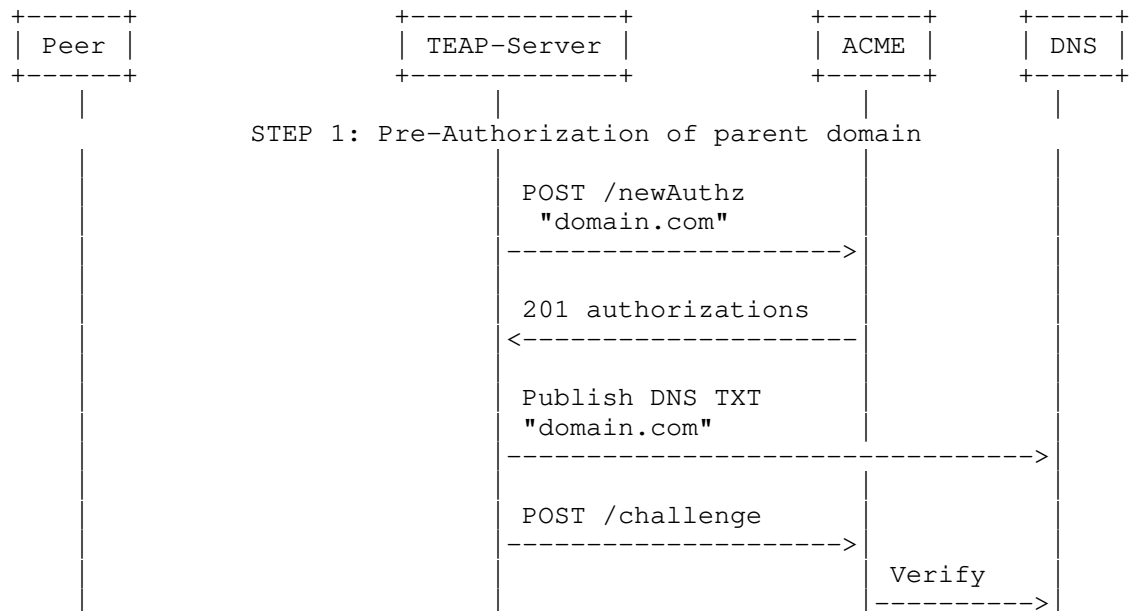
200 OK PKCS#7 "pledgeid.domain.com" <-----			
---	--	--	--

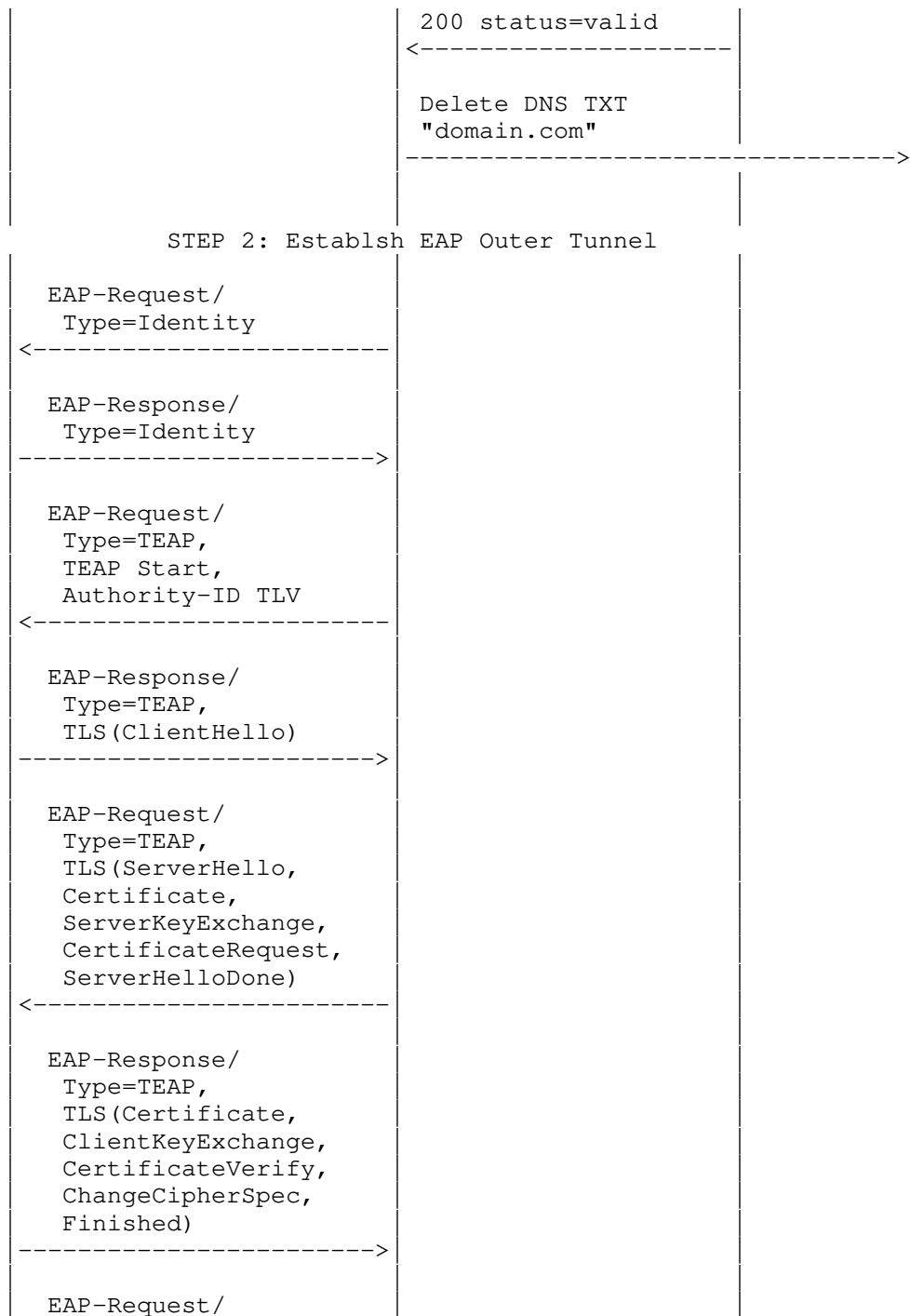
6. ACME Integration with TEAP

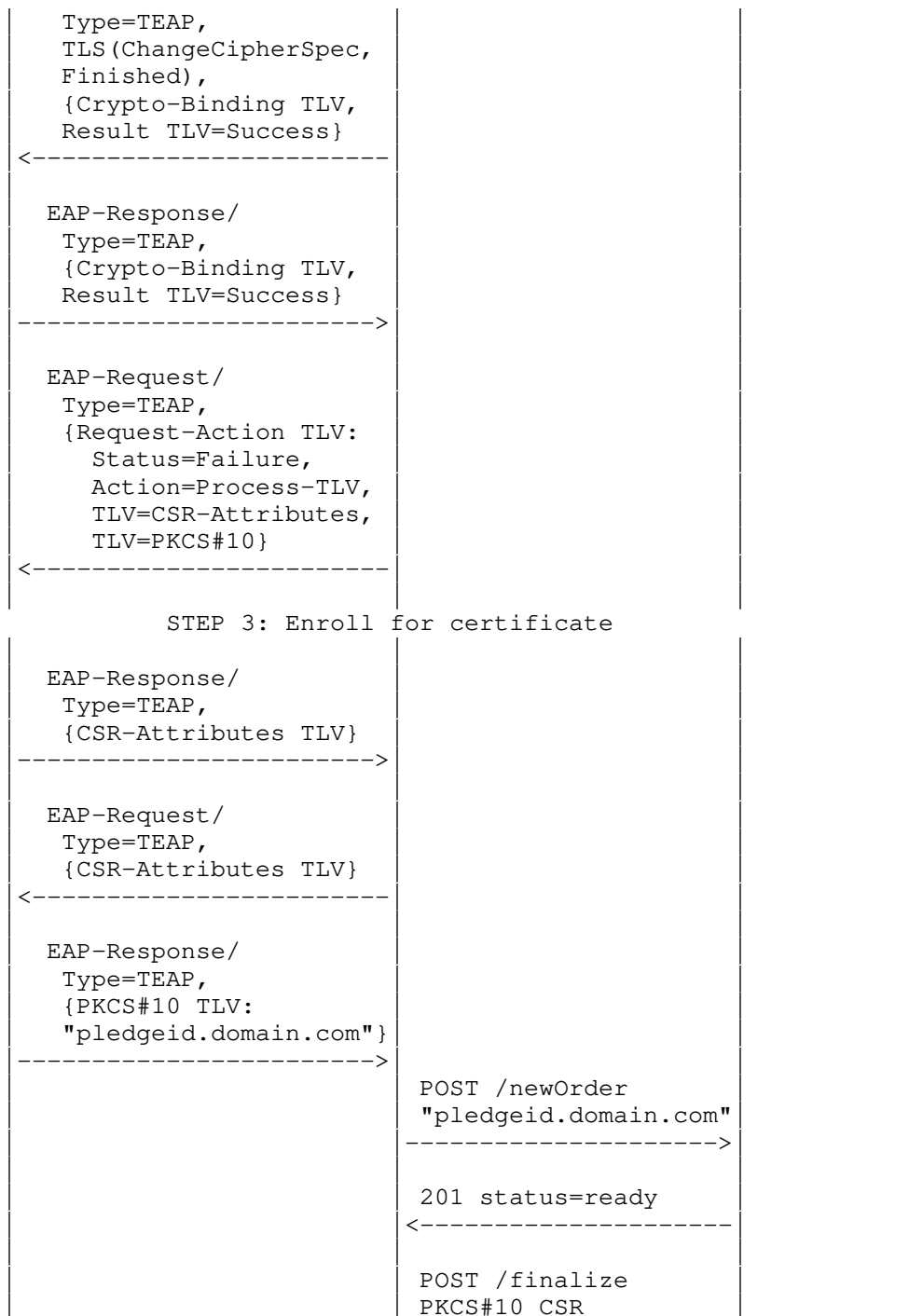
TEAP [RFC7170] defines a tunnel-based EAP method that enables secure communication between a peer and a server by using TLS to establish a mutually authenticated tunnel. TEAP enables certificate provisioning within the tunnel. TEAP does not define how the TEAP server communicates with the CA.

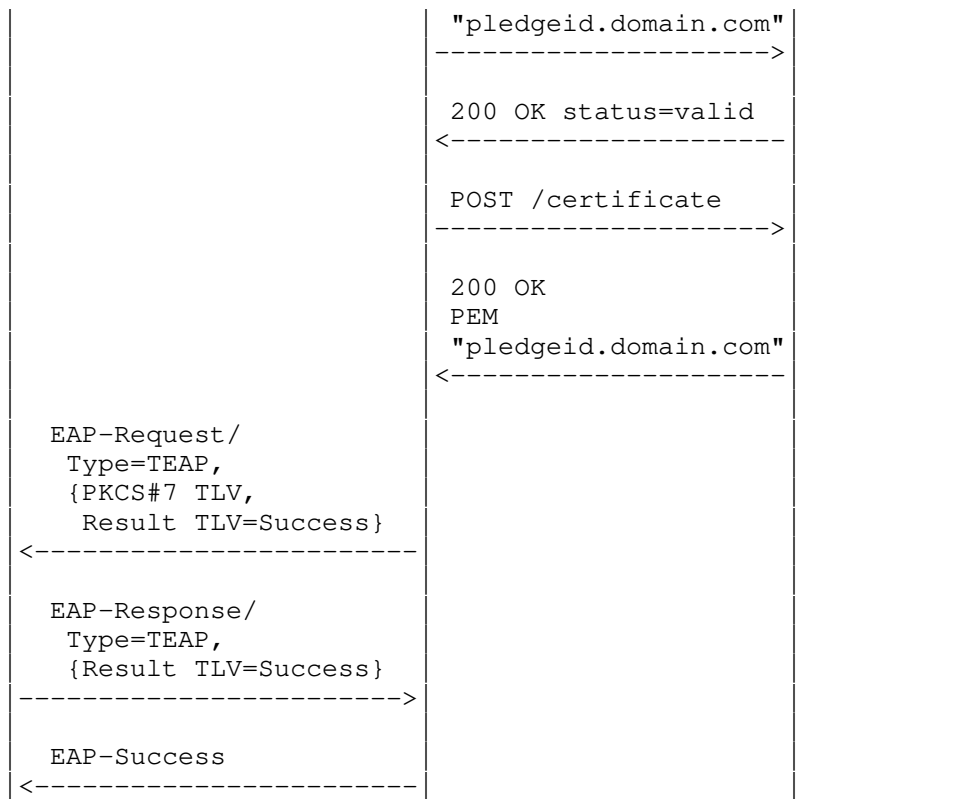
This section outlines how ACME could be used for communication between the TEAP server and the CA. The example call flow leverages [I-D.friel-acme-subdomains] and shows the TEAP server proving ownership of a parent domain, with individual client certificates being subdomains under that parent domain.

The example illustrates the TEAP server sending a Request-Action TLV including a CSR-Attributes TLV instructing the peer to send a CSR-Attributes TLV to the server. This enables the server to indicate what fields the peer should include in the CSR that the peer sends in the PKCS#10 TLV. For example, the TEAP server could instruct the peer what Subject or SAN entries to include in its CSR.







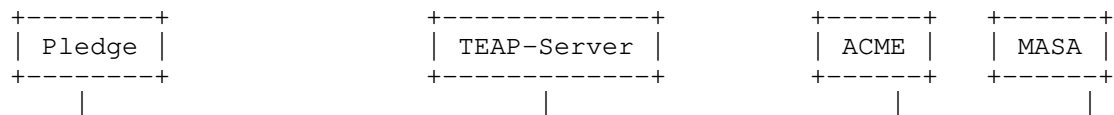


7. ACME Integration with TEAP-BRSKI

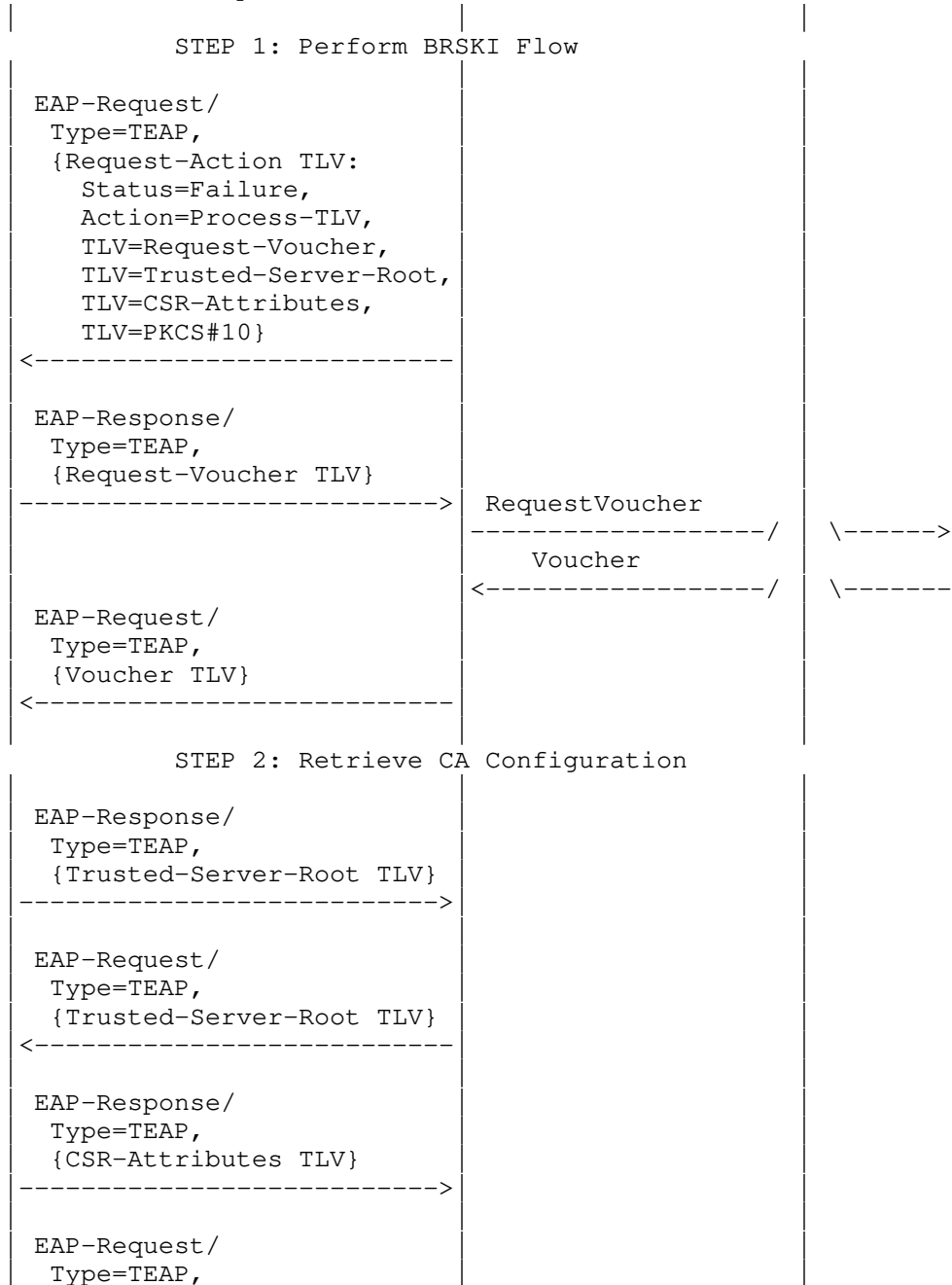
TEAP-BRSKI [I-D.lear-eap-teap-brski] defines how to execute BRSKI at layer 2 inside a TEAP tunnel. Similar to the TEAP proposal in the previous section, BRSKI-TEAP leverages the existing TEAP PKXS#10 and PKCS#7 mechanisms for certificate enrollment, and does not define how the TEAP server communicates with the CA.

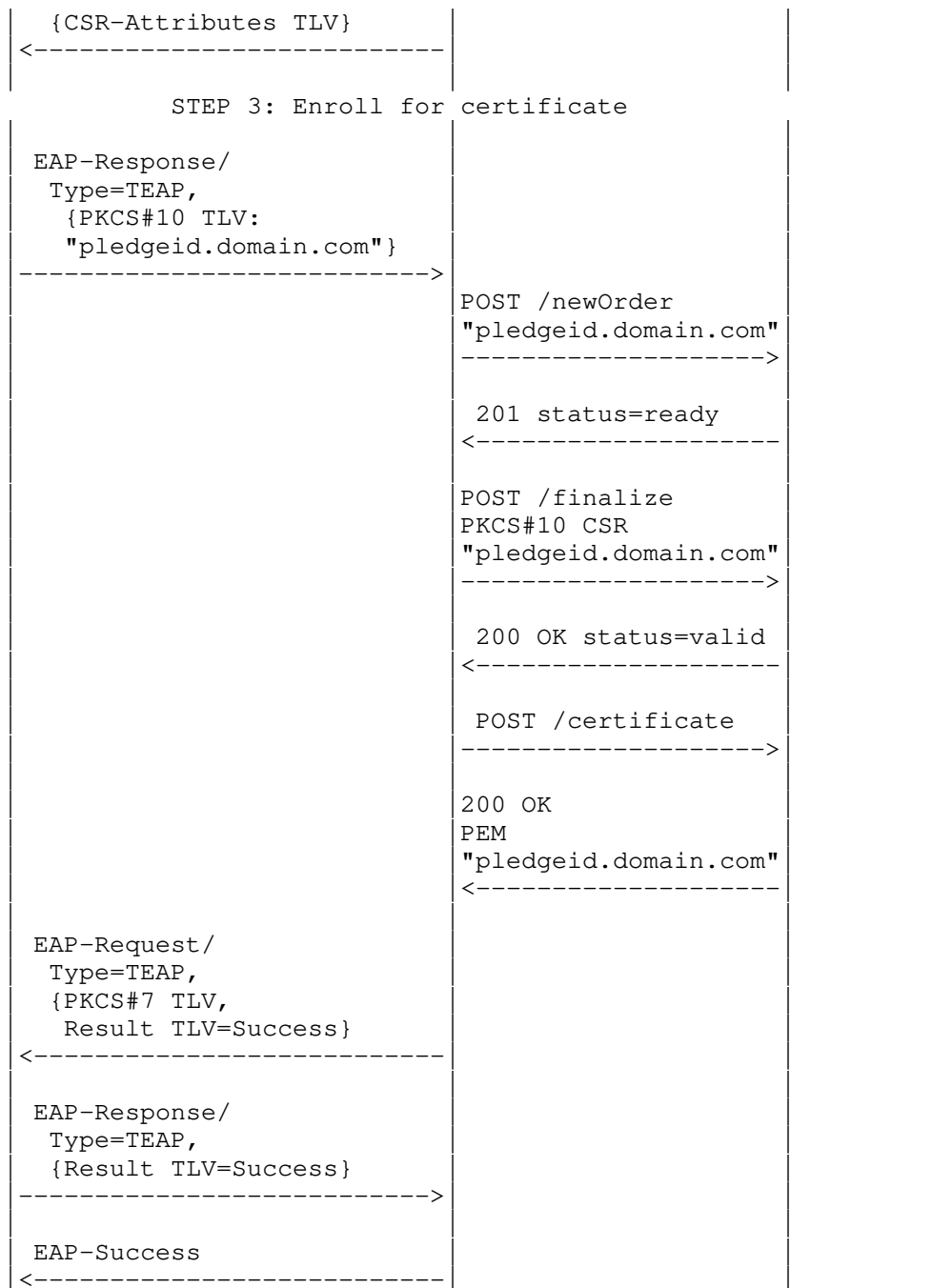
This section outlines how ACME could be used for communication between the TEAP server and the CA, and how this fits in with the TEAP-BRSKI proposal.

Similar to baseline TEAP, the TEAP server can use the CSR-Attributes TLV to tell the peer what attributes to include in its CSR request.



NOTE: Pre-Authorization of "domain.com" is complete and EAP outer tunnel is established as outlined in the previous section





8. IANA Considerations

[todo]

9. Security Considerations

[todo]

10. Informative References

[I-D.friel-acme-subdomains]

Friel, O., Barnes, R., and T. Hollebeek, "ACME for Subdomains", draft-friel-acme-subdomains-00 (work in progress), October 2019.

[I-D.friel-anima-brski-cloud]

Friel, O., Shekh-Yusef, R., and M. Richardson, "BRSKI Cloud Registrar", draft-friel-anima-brski-cloud-01 (work in progress), October 2019.

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

Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-28 (work in progress), September 2019.

[I-D.lear-eap-teap-brski]

Lear, E., Friel, O., Cam-Winget, N., and D. Harkins, "Bootstrapping Key Infrastructure over EAP", draft-lear-eap-teap-brski-04 (work in progress), September 2019.

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

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

[RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna, "Tunnel Extensible Authentication Protocol (TEAP) Version 1", RFC 7170, DOI 10.17487/RFC7170, May 2014, <<https://www.rfc-editor.org/info/rfc7170>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <<https://www.rfc-editor.org/info/rfc8555>>.

Appendix A. Comments

Authors' Addresses

Owen Friel
Cisco

Email: ofriel@cisco.com

Richard Barnes
Cisco

Email: rlb@ipv.sx

Rifaat Shekh-Yusef
Avaya

Email: rifaat.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 October 2021

O. Friel
Cisco
R. Shekh-Yusef
Auth0
M. Richardson
Sandelman Software Works
6 April 2021

BRSKI Cloud Registrar
draft-friel-anima-brski-cloud-04

Abstract

This document specifies the behaviour of a BRSKI Cloud Registrar, and how a pledge can interact with a BRSKI Cloud Registrar when bootstrapping.

RFCEd REMOVE: It is being actively worked on at <https://github.com/anima-wg/brski-cloud>

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 8 October 2021.

Copyright Notice

Copyright (c) 2021 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	2
1.1. Terminology	3
1.2. Target Use Cases	3
1.2.1. Owner Registrar Discovery	4
1.2.2. Bootstrapping with no Owner Registrar	4
2. Architecture	4
2.1. Interested Parties	5
2.2. Network Connectivity	6
2.3. Pledge Certificate Identity Considerations	6
3. Protocol Operation	6
3.1. Pledge Requests Voucher from Cloud Registrar	6
3.1.1. Cloud Registrar Discovery	6
3.1.2. Pledge - Cloud Registrar TLS Establishment Details	7
3.1.3. Pledge Issues Voucher Request	7
3.2. Cloud Registrar Handles Voucher Request	7
3.2.1. Pledge Ownership Lookup	8
3.2.2. Cloud Registrar Redirects to Owner Registrar	8
3.2.3. Cloud Registrar Issues Voucher	8
3.3. Pledge Handles Cloud Registrar Response	9
3.3.1. Redirect Response	9
3.3.2. Voucher Response	9
4. Protocol Details	9
4.1. Voucher Request Redirected to Local Domain Registrar	9
4.2. Voucher Request Handled by Cloud Registrar	11
5. YANG extension for Voucher based redirect	13
5.1. YANG Tree	13
5.2. YANG Voucher	14
6. IANA Considerations	16
7. Security Considerations	16
8. References	16
8.1. Normative References	16
8.2. Informative References	17
Authors' Addresses	17

1. Introduction

Bootstrapping Remote Secure Key Infrastructures (BRSKI) [I-D.ietf-anima-bootstrapping-keyinfra] specifies automated bootstrapping of an Autonomic Control Plane. BRSKI Section 2.7 describes how a 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".

This document further specifies use of a BRSKI cloud registrar and clarifies operations that are not sufficiently specified in BRSKI.

1.1. Terminology

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

This document uses the terms Pledge, Registrar, MASA, and Voucher from [I-D.ietf-anima-bootstrapping-keyinfra] and [RFC8366].

- * Local Domain: The domain where the pledge is physically located and bootstrapping from. This may be different to the pledge owner's domain.
- * Owner Domain: The domain that the pledge needs to discover and bootstrap with.
- * Cloud Registrar: The default Registrar that is deployed at a URI that is well known to the pledge.
- * Owner Registrar: The Registrar that is operated by the Owner, or the Owner's delegate. There may not be an Owner Registrar in all deployment scenarios.
- * Local Domain Registrar: The Registrar discovered on the Local Domain. There may not be a Local Domain Registrar in all deployment scenarios.

1.2. Target Use Cases

Two high level use cases are documented here. There are more details provided in sections Section 4.1 and Section 4.2. While both use cases aid with incremental deployment of BRSKI infrastructure, for many smaller sites (such as teleworkers) no further infrastructure are expected.

The pledge is not expected to know which of these two situations it is in. The pledge determines this based upon signals that it receives from the Cloud Registrar. The Cloud Registrar is expected to make the determination based upon the identity presented by the pledge.

While a Cloud Registrar will typically handle all the devices of a particular product line from a particular manufacturer there are no restrictions on how the Cloud Registrar is horizontally (many sites) or vertically (more equipment at one site) scaled. It is also entirely possible that all devices sold by through a particular VAR might be preloaded with a configuration that changes the Cloud Registrar URL to point to a VAR. Such an effort would require unboxing each device in a controlled environment, but the provisioning could occur using a regular BRSKI or SZTP [RFC8572] process.

1.2.1. Owner Registrar Discovery

A pledge is bootstrapping from a remote location with no local domain registrar (specifically: with no local infrastructure to provide for automated discovery), and needs to discover its owner registrar. The cloud registrar is used by the pledge to discover the owner registrar. The cloud registrar redirects the pledge to the owner registrar, and the pledge completes bootstrap against the owner registrar.

A typical example is an enduser deploying a pledge in a home or small branch office, where the pledge belongs to the enduser's employer. There is no local domain registrar, and the pledge needs to discover and bootstrap with the employer's registrar which is deployed in headquarters.

1.2.2. Bootstrapping with no Owner Registrar

A pledge is bootstrapping where the owner organization does not yet have an owner registrar deployed. The cloud registrar issues a voucher, and the pledge completes trust bootstrap using the cloud registrar. The voucher issued by the cloud includes domain information for the owner's EST [RFC7030] service the pledge should use for certificate enrollment.

In one use case, an organization has an EST service deployed, but does not have yet a BRSKI capable Registrar service deployed. The pledge is deployed in the organizations domain, but does not discover a local domain, or owner, registrar. The pledge uses the cloud registrar to bootstrap, and the cloud registrar provides a voucher that includes instructions on finding the organization's EST service.

2. Architecture

The high level architecture is illustrated in Figure 1.

The pledge connects to the cloud registrar during bootstrap.

The cloud registrar may redirect the pledge to an owner registrar in order to complete bootstrap against the owner registrar.

If the cloud registrar issues a voucher itself without redirecting the pledge to an owner registrar, the cloud registrar will inform the pledge what domain to use for accessing EST services in the voucher response.

Finally, when bootstrapping against an owner registrar, this registrar may interact with a backend CA to assist in issuing certificates to the pledge. The mechanisms and protocols by which the registrar interacts with the CA are transparent to the pledge and are out-of-scope of this document.

The architecture shows the cloud registrar and MASA as being logically separate entities. The two functions could of course be integrated into a single service.

TWO CHOICES: 1. Cloud Registrar redirects to Owner Registrar 2. Cloud Registrar returns VOUCHER pinning Owner Register.

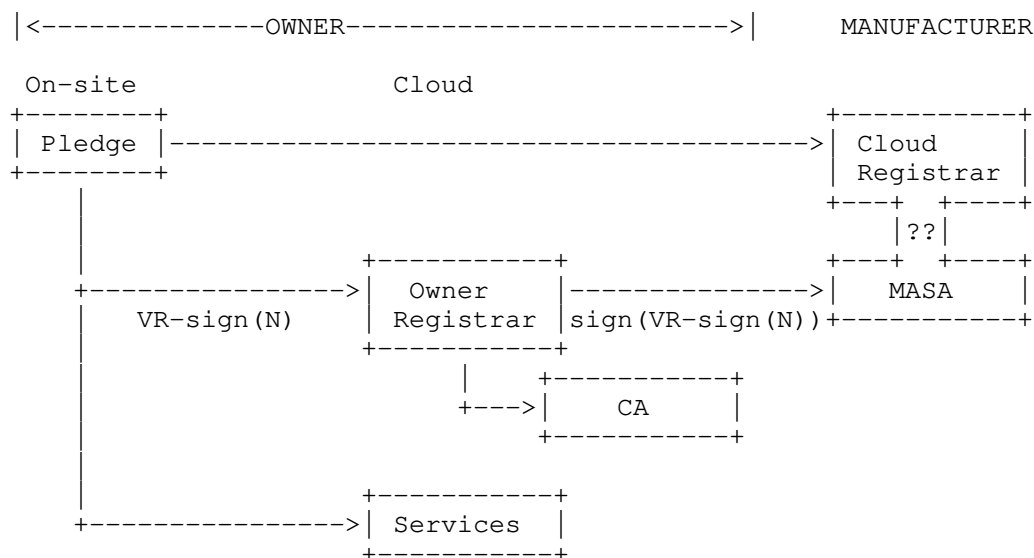


Figure 1: High Level Architecture

2.1. Interested Parties

1. OEM - Equipment manufacturer. Operate the MASA.

2. Network operator. Operate the Owner Registrar. Often operated by end owner (company), or by outsourced IT entity.
3. Network integrator. They operate a Cloud Registrar.

2.2. Network Connectivity

The assumption is that the pledge already has network connectivity prior to connecting to the cloud registrar. The pledge must have an IP address, must be able to make DNS queries, and must be able to send HTTP requests to the cloud registrar. The pledge operator has already connected the pledge to the network, and the mechanism by which this has happened is out of scope of this document.

2.3. Pledge Certificate Identity Considerations

BRSKI section 5.9.2 specifies that the pledge MUST send a CSR Attributes request to the registrar. The registrar MAY use this mechanism to instruct the pledge about the identities it should include in the CSR request it sends as part of enrollment. The registrar may use this mechanism to tell the pledge what Subject or Subject Alternative Name identity information to include in its CSR request. This can be useful if the Subject must have a specific value in order to complete enrollment with the CA.

For example, the pledge may only be aware of its IDevID Subject which includes a manufacturer serial number, but must include a specific fully qualified domain name in the CSR in order to complete domain ownership proofs required by the CA.

As another example, the registrar may deem the manufacturer serial number in an IDevID as personally identifiable information, and may want to specify a new random opaque identifier that the pledge should use in its CSR.

3. Protocol Operation

3.1. Pledge Requests Voucher from Cloud Registrar

3.1.1. Cloud Registrar Discovery

BRSKI defines how a pledge MAY contact a well known URI of a cloud registrar if a local domain registrar cannot be discovered. Additionally, certain pledge types may never attempt to discover a local domain registrar and may automatically bootstrap against a cloud registrar.

The details of the URI are manufacturer specific, with BRSKI giving the example "brski-registrar.manufacturer.example.com".

The Pledge SHOULD be provided with the entire URL of the Cloud Registrar, including the path component, which is typically "/.well-known/brski/requestvoucher", but may be another value.

3.1.2. Pledge - Cloud Registrar TLS Establishment Details

The pledge MUST use an Implicit Trust Anchor database (see [RFC7030]) to authenticate the cloud registrar service. The Pledge can be done with pre-loaded trust-anchors that are used to validate the TLS connection. This can be using a public Web PKI trust anchors using [RFC6125] DNS-ID mechanisms, a pinned certification authority, or even a pinned raw public key. This is a local implementation decision.

The pledge MUST NOT establish a provisional TLS connection (see BRSKI section 5.1) with the cloud registrar.

The cloud registrar MUST validate the identity of the pledge by sending a TLS CertificateRequest message to the pledge during TLS session establishment. The cloud registrar MAY include a certificate_authorities field in the message to specify the set of allowed IDevID issuing CAs that pledges may use when establishing connections with the cloud registrar.

The cloud registrar MAY only allow connections from pledges that have an IDevID that is signed by one of a specific set of CAs, e.g. IDevIDs issued by certain manufacturers.

The cloud registrar MAY allow pledges to connect using self-signed identity certificates or using Raw Public Key [RFC7250] certificates.

3.1.3. Pledge Issues Voucher Request

After the pledge has established a full TLS connection with the cloud registrar and has verified the cloud registrar PKI identity, the pledge generates a voucher request message as outlined in BRSKI section 5.2, and sends the voucher request message to the cloud registrar.

3.2. Cloud Registrar Handles Voucher Request

The cloud registrar must determine pledge ownership. Once ownership is determined, or if no owner can be determined, then the registrar may:

- * return a suitable 4xx or 5xx error response to the pledge if the registrar is unwilling or unable to handle the voucher request
- * redirect the pledge to an owner register via 307 response code
- * issue a voucher and return a 200 response code

3.2.1. Pledge Ownership Lookup

The cloud registrar needs some suitable mechanism for knowing the correct owner of a connecting pledge based on the presented identity certificate. For example, if the pledge establishes TLS using an IDevID that is signed by a known manufacturing CA, the registrar could extract the serial number from the IDevID and use this to lookup a database of pledge IDevID serial numbers to owners.

Alternatively, if the cloud registrar allows pledges to connect using self-signed certificates, the registrar could use the thumbprint of the self-signed certificate to lookup a database of pledge self-signed certificate thumbprints to owners.

The mechanism by which the cloud registrar determines pledge ownership is out-of-scope of this document.

3.2.2. Cloud Registrar Redirects to Owner Registrar

Once the cloud registrar has determined pledge ownership, the cloud registrar may redirect the pledge to the owner registrar in order to complete bootstrap. Ownership registration will require the owner to register their local domain. The mechanism by which pledge owners register their domain with the cloud registrar is out-of-scope of this document.

The cloud registrar replies to the voucher request with a suitable HTTP 307 response code, including the owner's local domain in the HTTP Location header.

3.2.3. Cloud Registrar Issues Voucher

If the cloud registrar issues a voucher, it returns the voucher in a HTTP response with a 200 response code.

The cloud registrar MAY issue a 202 response code if it is willing to issue a voucher, but will take some time to prepare the voucher.

The voucher MUST include the "est-domain" field as defined below. This tells the pledge where the domain of the EST service to use for completing certificate enrollment.

The voucher MAY include the "additional-configuration" field.. This points the pledge to a URI where application specific additional configuration information may be retrieved. Pledge and Registrar behavior for handling and specifying the "additional-configuration" field is out-of-scope of this document.

3.3. Pledge Handles Cloud Registrar Response

3.3.1. Redirect Response

The cloud registrar returned a 307 response to the voucher request. The pledge should complete BRSKI bootstrap as per standard BRSKI operation after following the HTTP redirect. The pledge should establish a provisional TLS connection with specified local domain registrar. The pledge should not use its Implicit Trust Anchor database for validating the local domain registrar identity. The pledge should send a voucher request message via the local domain registrar. When the pledge downloads a voucher, it can validate the TLS connection to the local domain registrar and continue with enrollment and bootstrap as per standard BRSKI operation.

3.3.2. Voucher Response

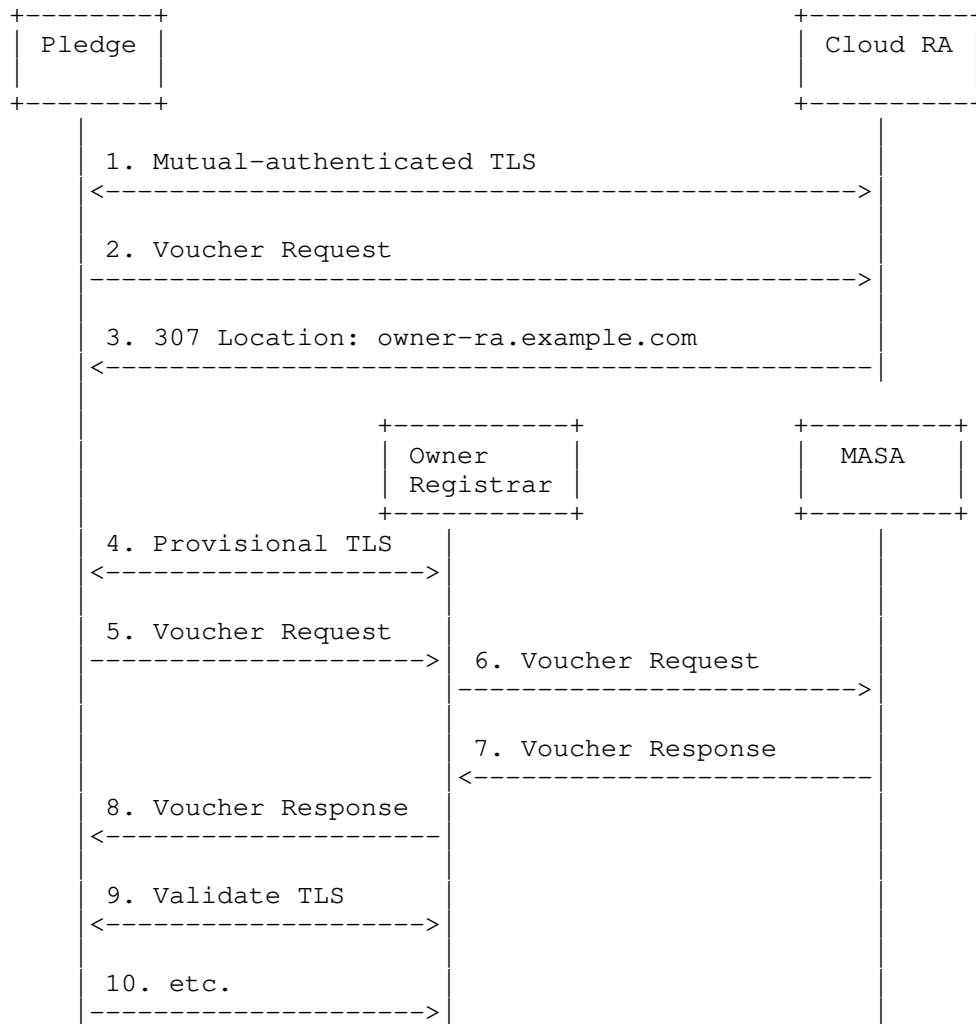
The cloud registrar returned a voucher to the pledge. The pledge should perform voucher verification as per standard BRSKI operation. The pledge should verify the voucher signature using the manufacturer-installed trust anchor(s), should verify the serial number in the voucher, and must verify any nonce information in the voucher.

The pledge should extract the "est-domain" field from the voucher, and should continue with EST enrollment as per standard BRSKI operation.

4. Protocol Details

4.1. Voucher Request Redirected to Local Domain Registrar

This flow illustrates the Owner Registrar Discovery flow. A pledge is bootstrapping in a remote location with no local domain registrar. The assumption is that the owner registrar domain is accessible and the pledge can establish a network connection with the owner registrar. This may require that the owner network firewall exposes the registrar on the public internet.



The process starts, in step 1, when the Pledge establishes a Mutual TLS channel with the Cloud RA using artifacts created during the manufacturing process of the Pledge.

In step 2, the Pledge sends a voucher request to the Cloud RA.

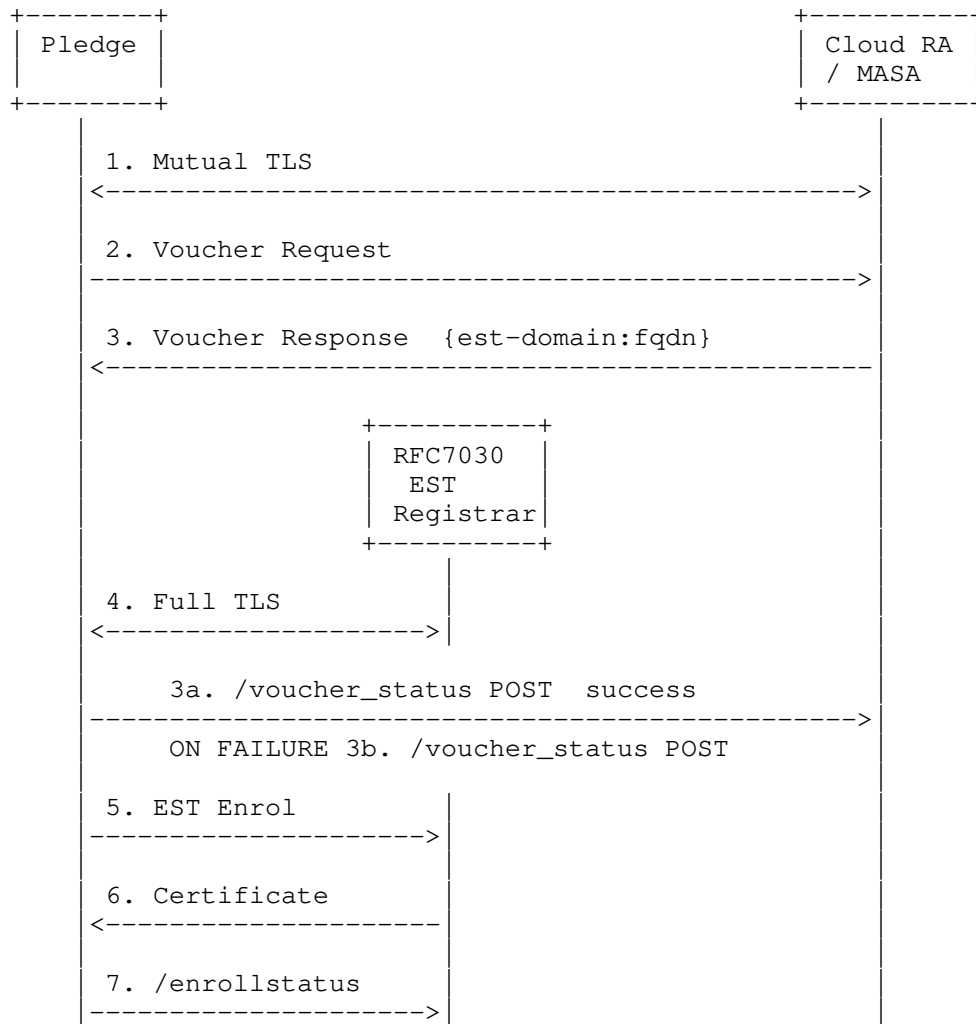
The Cloud RA completes pledge ownership lookup as outlined in Section 3.2.1, and determines the owner registrar domain. In step 3, the Cloud RA redirects the pledge to the owner registrar domain.

Steps 4 and onwards follow the standard BRSKI flow. The pledge establishes a provisional TLS connection with the owner registrar, and sends a voucher request to the owner registrar. The registrar forwards the voucher request to the MASA. Assuming the MASA issues a voucher, then the pledge validates the TLS connection with the registrar using the pinned-domain-cert from the voucher and completes the BRSKI flow.

4.2. Voucher Request Handled by Cloud Registrar

The Voucher includes the EST domain to use for EST enroll. It is assumed services are accessed at that domain too. As trust is already established via the Voucher, the pledge does a full TLS handshake against the local RA indicated by the voucher response.

The returned voucher contains an attribute, "est-domain", defined in Section 5 below. The pledge is directed to continue enrollment using the EST registrar found at that URI. The pledge uses the pinned-domain-cert from the voucher to authenticate the EST registrar.



The process starts, in step 1, when the Pledge establishes a Mutual TLS channel with the Cloud RA/MASA using artifacts created during the manufacturing process of the Pledge. In step 2, the Pledge sends a voucher request to the Cloud RA/MASA, and in response the Pledge receives an [RFC8366] format voucher from the Cloud RA/MASA that includes its assigned EST domain in the est-domain attribute.

At this stage, the Pledge should be able to establish a TLS channel with the EST Registrar. The connection may involve crossing the Internet requiring a DNS lookup on the provided name. It may also be a local address that includes an IP address literal including both [RFC1918] and IPv6 Unique Local Address. The EST Registrar is

validated using the pinned-domain-cert value provided in the voucher as described in section 5.6.2 of [I-D.ietf-anima-bootstrapping-keyinfra]. This involves treating the artifact provided in the pinned-domain-cert as a trust anchor, and attempting to validate the EST Registrar from this anchor only.

There is a case where the pinned-domain-cert is the identical End-Entity (EE) Certificate as the EST Registrar. It also explicitly includes the case where the EST Registrar has a self-signed EE Certificate, but it may also be an EE certificate that is part of a larger PKI. If the certificate is not a self-signed or EE certificate, then the Pledge SHOULD apply [RFC6125] DNS-ID validation on the certificate against the URL provided in the est-domain attribute. If the est-domain was provided by with an IP address literal, then it is unlikely that it can be validated, and in that case, it is expected that either a self-signed certificate or an EE certificate will be pinned.

The Pledge also has the details it needs to be able to create the CSR request to send to the RA based on the details provided in the voucher.

In step 4, the Pledge establishes a TLS channel with the Cloud RA/MASA, and optionally the pledge should send a request, steps 3.a and 3.b, to the Cloud RA/MASA to inform it that the Pledge was able to establish a secure TLS channel with the EST Registrar.

The Pledge then follows that, in step 5, with an EST Enroll request with the CSR and obtains the requested certificate. The Pledge must validate that the issued certificate has the expected identifier obtained from the Cloud RA/MASA in step 3.

5. YANG extension for Voucher based redirect

An extension to the [RFC8366] voucher is needed for the case where the client will be redirected to a local EST Registrar.

5.1. YANG Tree

```
module: ietf-redirected-voucher

  grouping voucher-redirected-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
      +-- est-domain?               ietf:uri
      +-- additional-configuration? ietf:uri
```

5.2. YANG Voucher

```
<CODE BEGINS> file "ietf-redirected-voucher@2020-09-23.yang"
module ietf-redirected-voucher {
  yang-version 1.1;

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

  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-inet-types {
    prefix ietf;
    reference "RFC 6991: Common YANG Data Types";
  }

  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: Owen Friel
 <mailto:ofriel@cisco.com>
 Author: Rifaat Shekh-Yusef
 <mailto:rifaat.ietf@gmail.com>;

description

"This module extends the base RFC8366 voucher format to include a redirect to an EST server to which enrollment should continue.

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 BCP14, RFC 2119, and RFC8174.";

```
revision "2020-09-23" {
  description
    "Initial version";
  reference
    "RFC XXXX: Voucher Profile for Cloud redirected Devices";
}
```

```
rc:yang-data voucher-redirected-artifact {
  // YANG data template for a voucher.
  uses voucher-redirected-grouping;
}
```

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

```
  uses v:voucher-artifact-grouping {

    augment "voucher" {
      description "Base the constrained voucher
                  upon the regular one";
      leaf est-domain {
        type ietf:uri;
        description
          "The est-domain is a URL to which the Pledge should continue
           doing enrollment rather than with the Cloud Registrar.";
      }
      leaf additional-configuration {
        type ietf:uri;
        description
          "The additional-configuration attribute contains a URL to which the
           Pledge can retrieve additional configuration
```

```
        information. The contents of this URL are vendor specific. This
is intended to do things like configure
        a VoIP phone to point to the correct hosted PBX, for example.";
    }
}
}
}
}
<CODE ENDS>
```

6. IANA Considerations

TODO:MCR - Will need to add IETF YANG registration from templates. [[
TODO]]

7. Security Considerations

[[TODO]]

8. References

8.1. Normative References

- [I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M. C., Eckert, T., Behringer, M.
H., and K. Watsen, "Bootstrapping Remote Secure Key
Infrastructures (BRSKI)", Work in Progress, Internet-
Draft, draft-ietf-anima-bootstrapping-keyinfra-45, 11
November 2020, <[https://www.ietf.org/internet-drafts/
draft-ietf-anima-bootstrapping-keyinfra-45.txt](https://www.ietf.org/internet-drafts/draft-ietf-anima-bootstrapping-keyinfra-45.txt)>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
"Enrollment over Secure Transport", RFC 7030,
DOI 10.17487/RFC7030, October 2013,
<<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,
"A Voucher Artifact for Bootstrapping Protocols",
RFC 8366, DOI 10.17487/RFC8366, May 2018,
<<https://www.rfc-editor.org/info/rfc8366>>.

8.2. Informative References

- [IEEE802.1AR]
IEEE Standard, ., "IEEE 802.1AR Secure Device Identifier",
2018, <[http://standards.ieee.org/findstds/
standard/802.1AR-2018.html](http://standards.ieee.org/findstds/standard/802.1AR-2018.html)>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.
J., and E. Lear, "Address Allocation for Private
Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918,
February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
Verification of Domain-Based Application Service Identity
within Internet Public Key Infrastructure Using X.509
(PKIX) Certificates in the Context of Transport Layer
Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
Weiler, S., and T. Kivinen, "Using Raw Public Keys in
Transport Layer Security (TLS) and Datagram Transport
Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
June 2014, <<https://www.rfc-editor.org/info/rfc7250>>.
- [RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero
Touch Provisioning (SZTP)", RFC 8572,
DOI 10.17487/RFC8572, April 2019,
<<https://www.rfc-editor.org/info/rfc8572>>.

Authors' Addresses

Owen Friel
Cisco

Email: ofriel@cisco.com

Rifaat Shekh-Yusef
Auth0

Email: rifaat.s.ietf@gmail.com

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: 3 May 2021

T. Eckert, Ed.
Futurewei USA
M. Behringer, Ed.

S. Bjarnason
Arbor Networks
30 October 2020

An Autonomic Control Plane (ACP)
draft-ietf-anima-autonomic-control-plane-30

Abstract

Autonomic functions need a control plane to communicate, which depends on some addressing and routing. This Autonomic 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 3 May 2021.

Copyright Notice

Copyright (c) 2020 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)	6
1.1. Applicability and Scope	9
2. Acronyms and Terminology (Informative)	11
3. Use Cases for an Autonomic Control Plane (Informative)	16
3.1. An Infrastructure for Autonomic Functions	17
3.2. Secure Bootstrap over a not configured Network	17
3.3. Data-Plane Independent Permanent Reachability	17
4. Requirements (Informative)	19
5. Overview (Informative)	20
6. Self-Creation of an Autonomic Control Plane (ACP) (Normative)	21
6.1. Requirements for use of Transport Layer Security (TLS)	22
6.2. ACP Domain, Certificate and Network	23
6.2.1. ACP Certificates	24
6.2.2. ACP Certificate AcpNodeName	26
6.2.2.1. AcpNodeName ASN.1 Module	29
6.2.3. ACP domain membership check	30
6.2.3.1. Realtime clock and Time Validation	33
6.2.4. Trust Anchors (TA)	33
6.2.5. Certificate and Trust Anchor Maintenance	34
6.2.5.1. GRASP objective for EST server	35
6.2.5.2. Renewal	37
6.2.5.3. Certificate Revocation Lists (CRLs)	37
6.2.5.4. Lifetimes	38
6.2.5.5. Re-enrollment	38
6.2.5.6. Failing Certificates	40
6.3. ACP Adjacency Table	41
6.4. Neighbor Discovery with DULL GRASP	41
6.5. Candidate ACP Neighbor Selection	45
6.6. Channel Selection	45
6.7. Candidate ACP Neighbor verification	49
6.8. Security Association (Secure Channel) protocols	49
6.8.1. General considerations	50
6.8.2. Common requirements	51
6.8.3. ACP via IPsec	52
6.8.3.1. Native IPsec	52
6.8.3.1.1. RFC8221 (IPsec/ESP)	53

6.8.3.1.2. RFC8247 (IKEv2)	54
6.8.3.2. IPsec with GRE encapsulation	55
6.8.4. ACP via DTLS	56
6.8.5. ACP Secure Channel Profiles	58
6.9. GRASP in the ACP	59
6.9.1. GRASP as a core service of the ACP	59
6.9.2. ACP as the Security and Transport substrate for GRASP	59
6.9.2.1. Discussion	62
6.10. Context Separation	63
6.11. Addressing inside the ACP	63
6.11.1. Fundamental Concepts of Autonomic Addressing	63
6.11.2. The ACP Addressing Base Scheme	65
6.11.3. ACP Zone Addressing Sub-Scheme (ACP-Zone)	67
6.11.4. ACP Manual Addressing Sub-Scheme (ACP-Manual)	68
6.11.5. ACP Vlong Addressing Sub-Scheme (ACP-VLong-8/ ACP-VLong-16	69
6.11.6. Other ACP Addressing Sub-Schemes	70
6.11.7. ACP Registrars	71
6.11.7.1. Use of BRSKI or other Mechanism/Protocols	71
6.11.7.2. Unique Address/Prefix allocation	72
6.11.7.3. Addressing Sub-Scheme Policies	72
6.11.7.4. Address/Prefix Persistence	74
6.11.7.5. Further Details	74
6.12. Routing in the ACP	74
6.12.1. ACP RPL Profile	75
6.12.1.1. Overview	75
6.12.1.1.1. Single Instance	75
6.12.1.1.2. Reconvergence	76
6.12.1.2. RPL Instances	77
6.12.1.3. Storing vs. Non-Storing Mode	77
6.12.1.4. DAO Policy	77
6.12.1.5. Path Metric	77
6.12.1.6. Objective Function	77
6.12.1.7. DODAG Repair	77
6.12.1.8. Multicast	78
6.12.1.9. Security	78
6.12.1.10. P2P communications	78
6.12.1.11. IPv6 address configuration	78
6.12.1.12. Administrative parameters	79
6.12.1.13. RPL Packet Information	79
6.12.1.14. Unknown Destinations	79
6.13. General ACP Considerations	80
6.13.1. Performance	80
6.13.2. Addressing of Secure Channels	80
6.13.3. MTU	81
6.13.4. Multiple links between nodes	81
6.13.5. ACP interfaces	82

6.13.5.1.	ACP loopback interfaces	82
6.13.5.2.	ACP virtual interfaces	84
6.13.5.2.1.	ACP point-to-point virtual interfaces	84
6.13.5.2.2.	ACP multi-access virtual interfaces	84
7.	ACP support on L2 switches/ports (Normative)	87
7.1.	Why (Benefits of ACP on L2 switches)	87
7.2.	How (per L2 port DULL GRASP)	88
8.	Support for Non-ACP Components (Normative)	89
8.1.	ACP Connect	89
8.1.1.	Non-ACP Controller / NMS system	90
8.1.2.	Software Components	92
8.1.3.	Auto Configuration	93
8.1.4.	Combined ACP/Data-Plane Interface (VRF Select)	94
8.1.5.	Use of GRASP	96
8.2.	Connecting ACP islands over Non-ACP L3 networks (Remote ACP neighbors)	97
8.2.1.	Configured Remote ACP neighbor	97
8.2.2.	Tunneled Remote ACP Neighbor	98
8.2.3.	Summary	98
9.	ACP Operations (Informative)	99
9.1.	ACP (and BRSKI) Diagnostics	99
9.1.1.	Secure Channel Peer diagnostics	103
9.2.	ACP Registrars	104
9.2.1.	Registrar interactions	104
9.2.2.	Registrar Parameter	105
9.2.3.	Certificate renewal and limitations	106
9.2.4.	ACP Registrars with sub-CA	107
9.2.5.	Centralized Policy Control	107
9.3.	Enabling and disabling ACP/ANI	108
9.3.1.	Filtering for non-ACP/ANI packets	108
9.3.2.	Admin Down State	109
9.3.2.1.	Security	110
9.3.2.2.	Fast state propagation and Diagnostics	110
9.3.2.3.	Low Level Link Diagnostics	111
9.3.2.4.	Power Consumption Issues	112
9.3.3.	Interface level ACP/ANI enable	112
9.3.4.	Which interfaces to auto-enable?	112
9.3.5.	Node Level ACP/ANI enable	114
9.3.5.1.	Brownfield nodes	114
9.3.5.2.	Greenfield nodes	115
9.3.6.	Undoing ANI/ACP enable	116
9.3.7.	Summary	117
9.4.	Partial or Incremental adoption	117
9.5.	Configuration and the ACP (summary)	118
10.	Summary: Benefits (Informative)	119
10.1.	Self-Healing Properties	119
10.2.	Self-Protection Properties	121
10.2.1.	From the outside	121

10.2.2. From the inside	122
10.3. The Administrator View	123
11. Security Considerations	124
12. IANA Considerations	129
13. Acknowledgements	130
14. Contributors	130
15. Change log [RFC-Editor: Please remove]	131
15.1. Summary of changes since entering IESG review	131
15.1.1. Reviews (while in IESG review status) / status	131
15.1.2. BRSKI / ACP registrar related enhancements	132
15.1.3. Normative enhancements since start of IESG review	132
15.1.4. Explanatory enhancements since start of IESG review	133
15.2. draft-ietf-anima-autonomic-control-plane-30	134
15.3. draft-ietf-anima-autonomic-control-plane-29	136
15.4. draft-ietf-anima-autonomic-control-plane-28	138
15.5. draft-ietf-anima-autonomic-control-plane-27	140
15.6. draft-ietf-anima-autonomic-control-plane-26	140
15.7. draft-ietf-anima-autonomic-control-plane-25	141
15.8. draft-ietf-anima-autonomic-control-plane-24	144
15.9. draft-ietf-anima-autonomic-control-plane-23	145
15.10. draft-ietf-anima-autonomic-control-plane-22	146
16. Normative References	148
17. Informative References	151
Appendix A. Background and Futures (Informative)	160
A.1. ACP Address Space Schemes	160
A.2. BRSKI Bootstrap (ANI)	161
A.3. ACP Neighbor discovery protocol selection	162
A.3.1. LLDP	162
A.3.2. mDNS and L2 support	163
A.3.3. Why DULL GRASP	163
A.4. Choice of routing protocol (RPL)	163
A.5. ACP Information Distribution and multicast	165
A.6. CAs, domains and routing subdomains	166
A.7. Intent for the ACP	167
A.8. Adopting ACP concepts for other environments	168
A.9. Further (future) options	170
A.9.1. Auto-aggregation of routes	170
A.9.2. More options for avoiding IPv6 Data-Plane dependencies	170
A.9.3. ACP APIs and operational models (YANG)	171
A.9.4. RPL enhancements	171
A.9.5. Role assignments	172
A.9.6. Autonomic L3 transit	172
A.9.7. Diagnostics	172
A.9.8. Avoiding and dealing with compromised ACP nodes	173
A.9.9. Detecting ACP secure channel downgrade attacks	174

Appendix B. Unfinished considerations (To Be Removed From RFC)	175
B.1. Considerations for improving secure channel negotiation	175
B.2. ACP address verification	176
B.3. Public CA considerations	178
B.4. Hardening DULL GRASP considerations	179
Authors' Addresses	179

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 OAM and Control". For naming consistency with that prior document, this document continues to use the name ACP though.

Today, the OAM and control plane of IP networks is what is typically called in-band management/signaling: Its management and control protocol traffic depends on the routing and forwarding tables, security, policy, QoS and potentially other configuration that first has to be established through the very same management and control protocols. Misconfigurations including unexpected side effects or mutual dependences can disrupt OAM and control operations and especially disrupt remote management access to the affected node itself and potentially a much larger number of nodes for whom the affected node is on the network path.

For an example of inband management failing in the face of operator induced misconfiguration, see [FCC], for example III.B.15 on page 8: "...engineers almost immediately recognized that they had misdiagnosed the problem. However, they were unable to resolve the issue by restoring the link because the network management tools required to do so remotely relied on the same paths they had just disabled".

Traditionally, physically separate, so-called out-of-band (management) networks have been used to avoid these problems or at least to allow recovery from such problems. Worst case, 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 ACP, which is a virtual out-of-band network designed to be as independent as possible of configuration, addressing and routing to avoid the self-dependency problems of current IP networks while still operating in-band on the same physical network that it is controlling and managing. The ACP design is therefore intended to combine as well as possible the resilience of out-of-band management networks with the low-cost of traditional IP in-band network management. The details how this is achieved are described in Section 6.

In a fully autonomic network node without legacy control or management functions/protocols, the Data-Plane would be for example just a forwarding plane for "Data" IPv6 packets, aka: packets other than the control and management plane packets that are forwarded by the ACP itself. In such networks/nodes, there would be no non-autonomous control or non-autonomous management plane.

Routing protocols for example would be built inside the ACP as so-called autonomous functions via autonomous service agents, leveraging the ACP's functions instead of implementing them separately for each protocol: discovery, automatically established authenticated and encrypted local and distant peer connectivity for control and management traffic, and common control/management protocol session and presentation functions.

When ACP functionality is added to nodes that have non-autonomous management plane and/or control plane functions (henceforth called non-autonomous nodes), the ACP instead is best abstracted as a special Virtual Routing and Forwarding (VRF) instance (or virtual router) and the complete pre-existing non-autonomous management and/or control plane is considered to be part of the Data-Plane to avoid introduction of more complex, new terminology only for this case.

Like the forwarding plane for "Data" packets, the non-autonomous control and management plane functions can then be managed/used via the ACP. This terminology is consistent with pre-existing documents such as [RFC8368].

In both instances (autonomous and non-autonomous nodes), the ACP is built such that it is operating in the absence of the Data-Plane, and in the case of existing non-autonomous (management, control) components in the Data-Plane also in the presence of any (mis-)configuration thereof.

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 access remote devices using protocols such as Secure SHell (SSH) or Network Configuration Protocol (NETCONF) running across the ACP, 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 of how the ACP is constructed.

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

The remaining sections are non-normative: Section 10 reviews benefits of the ACP (after all the details have been defined), Section 9 provides operational recommendations, Appendix A provides additional explanations and describes additional details or future standard or proprietary 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 6.4) to autodiscover ACP neighbors, and includes the ACP GRASP instance to provide service discovery for clients of the ACP (see Section 6.9) 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 OAM applications, specifically for the case of current non-autonomic networks/nodes. 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" (ANI) as defined in [I-D.ietf-anima-reference-model], which provides autonomic connectivity (from ACP) with 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), Metro(politan 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, integrity-protection 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 (DTLS, see Section 6.8.4).

The implementation footprint of the ACP consists of Public Key Infrastructure (PKI) code for the ACP certificate including "Enrollment over Secure Transport (EST, see [RFC7030]), the GRASP protocol, UDP, TCP and Transport Layer Security (TLS, see Section 6.1), for security and reliability of GRASP and for EST, 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 and it could continue to be IPv4 only. For such IPv4-only devices, the IPv6 protocol itself would be additional implementation footprint that is only required 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 devices constrained with respect to cpu and memory, and to networks constrained with respect to bitrate and reliability, 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 6.9.2) only specifies TCP/TLS. See Appendix A.8 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: Please add ACP, BRSKI, GRASP, MASA to <https://www.rfc-editor.org/materials/abbrev.expansion.txt>.]

[RFC-Editor: What is the recommended way to reference a hanging text, e.g. to a definition in the list of definitions? Up to -28, this document was using XMLv2 and the only option I could find for RFC/XML to point to a hanging text was `format="title"`, which leads to references such as `'->"ACP certificate" ()'`, aka: redundant empty parenthesis. Many reviewers were concerned about this. I 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>. When I changed to XMLv3 in version -29, I could get rid of the unnecessary `()` by using `format="none"`, but that format is declared to be deprecated in XMLv3. So I am not aware of any working AND "non-deprecated" option.]

[RFC-Editor: Question: Is it possible to change the first occurrences of [RFCxxxx] references to "rfcxxx 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.]

This document serves both as a normative specification for how ACP nodes have to behave as well as describing requirements, benefits, architecture and operational aspects to explain the context. Normative sections are labelled "(Normative)" and use BCP 14 keywords. Other sections are labelled "(Informative)" and do not use those normative keywords.

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 acp-node-name of the ->"ACP 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 certificates" that allow them to authenticate each other as members of the ACP domain. See also Section 6.2.3.
- ACP (ANI/AN) certificate: A [RFC5280] certificate (LDevID) carrying the acp-node-name 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.
- ACP acp-node-name field: An information field in the ACP certificate in which the ACP relevant information is encoded: the ACP domain name, the ACP IPv6 address of the node and optional additional role attributes about the node.
- ACP Loopback interface: The Loopback interface in the ACP Virtual Routing and Forwarding (VRF) that has the ACP address assigned to it. See Section 6.13.5.1.
- 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 6.11. 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 6.2.2. 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 channel authenticated via ->"ACP certificates"

providing integrity protection and confidentiality through encryption. These are established between (normally) adjacent ACP nodes to carry traffic of the ACP VRF securely 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 6.13.5.
- 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 acp-node-name of the Domain Certificate. See Section 6.2.2.
- 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 specifications 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.
- CA: "Certification Authority". An entity that issues digital certificates. A CA uses its private key to sign the certificates it issues. Relying parties use the public key in the CA certificate to validate the signature.
- CRL: "Certificate Revocation List". A list of revoked certificates. Required to revoke certificates before their lifetime expires.
- Data-Plane: The counterpoint to the ACP VRF in an ACP node: forwarding of user traffic and in non-autonomous nodes/networks also any non-autonomous control and/or management plane functions. In a fully Autonomic Network node, the Data-Plane is managed autonomically via Autonomic Functions and Intent. See Section 1 for more detailed explanations.
- device: A physical system, or physical node.

Enrollment: The process through which a node authenticates itself to a network with an initial identity, which is often called IDevID certificate, and acquires from the network a network specific identity, which is often called LDevID certificate, and certificates of one or more Trust Anchor(s). In the ACP, the LDevID certificate is called the ACP certificate.

EST: "Enrollment over Secure Transport" ([RFC7030]). IETF standard-track protocol for enrollment of a node with an LDevID certificate. 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]. The IDevID certificate cannot be used as a node identifier 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/signaling): In-band management traffic and/or control plane signaling uses the same network resources such as routers/switches and network links that it manages/controls. In-band is the standard management and signaling mechanism in IP networks. Compared to ->"out-of-band" it requires no additional physical resources, but introduces potentially circular dependencies for its correct operations. See ->"introduction".

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

Loopback interface: See ->"ACP Loopback interface".

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

Management: Used in this document as another word for ->"OAM".

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

MIC: "Manufacturer Installed Certificate". This is another word to describe an IDevID in referenced materials. This term is not used in this document.

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.

NOC: Network Operations Center.

node: A system 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 6.11.3) or 78-bits (see Section 6.11.5) of the ACP address.

OAM: Operations, Administration and Management. Includes Network Monitoring.

Operational Technology (OT): https://en.wikipedia.org/wiki/Operational_Technology: "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.

out-of-band (management) 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. See ->"Introduction"

(virtual) out-of-band network: The ACP can be called a virtual out-of-band network for management and control because it attempts to provide the benefits of a (physical) ->"out-of-band network" even though it is physically carried ->"in-band". See ->"introduction".

root CA: "root Certification Authority". A ->"CA" for which the root CA Key update procedures of [RFC7030], Section 4.4 can be applied.

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

(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 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 6.11.7. Renewal and other maintenance (such as revocation) of ACP certificates may be performed by other entities than registrars. EST must be supported for ACP certificate renewal (see Section 6.2.5). BRSKI is an extension of EST, so ANI/BRSKI registrars can easily support ACP domain certificate renewal in addition to initial enrollment.

RPI: "RPL Packet Information". Network extension headers for use with the ->"RPL" routing protocols. Not used with RPL in the ACP. See Section 6.12.1.13.

RPL: "Routing Protocol for Low-Power and Lossy Networks". The routing protocol used in the ACP. See Section 6.12.

- sUDI: "secured Unique Device Identifier". This is another word to describe an IDevID in referenced material. This term is not used in this document.
- TA: "Trust Anchor". A Trust Anchor is an entity that is trusted for the purpose of certificate validation. Trust Anchor Information such as self-signed certificate(s) of the Trust Anchor is configured into the ACP node as part of Enrollment. See [RFC5280], Section 6.1.1.
- 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]. ULA is the IPv6 successor of the IPv4 private address space ([RFC1918]). ULA have important differences over IPv4 private addresses that are beneficial for and exploited by the ACP, such as the Locally Assigned Global ID prefix, which are the first 48-bits of a ULA address [RFC4193], section 3.2.1. In this document this prefix 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 6.11.3. 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)

This section summarizes the use cases that are intended to be supported by an ACP. To understand how these are derived from and relate to the larger set of use cases for autonomic networks, please refer to [RFC8316].

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 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 OAM protocols are using the Data-Plane of the network. This leads to often undesirable dependencies between control and OAM 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 OAM/control 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 6.13.2 for more details), the ACP provides reachability that is independent of the Data-Plane. This allows the control plane and OAM 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 uses Unique Local Addresses (ULA) for this purpose, 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 it is very strongly > recommended that they be encrypted.

Explanation for ACP4: In a fully autonomic network (AN), newly written ASAs 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)

When a node has an ACP certificate (see Section 6.2.1) and is enabled to bring up the ACP (see Section 9.3.5), it will create its ACP without any configuration as follows. For details, see Section 6 and further sections:

1. The node creates a VRF instance, or a similar virtual context for the ACP.
2. The node assigns its ULA IPv6 address (prefix) (see Section 6.11 which is learned from the `acp-node-name` (see Section 6.2.2) of its ACP certificate (see Section 6.2.1) to an ACP loopback interface (see Section 6.11) and connects this interface into the ACP VRF.
3. The node establishes a list of candidate peer adjacencies and candidate channel types to try for the adjacency. This is automatic for all candidate link-local adjacencies, see Section 6.4 across all native interfaces (see Section 9.3.4). If a candidate peer is discovered via multiple interfaces, this will result in one adjacency per interface. If the ACP node has multiple interfaces connecting to the same subnet across which it is also operating as an L2 switch in the Data-Plane, it employs methods for ACP with L2 switching, see Section 7.
4. For each entry in the candidate adjacency list, the node negotiates a secure tunnel using the candidate channel types. See Section 6.6.
5. The node authenticates the peer node during secure channel setup and authorizes it to become part of the ACP according to Section 6.2.3.
6. Unsuccessful authentication of a candidate peer results in throttled connection retries for as long as the candidate peer is discoverable. See Section 6.7.
7. Each successfully established secure channel is mapped into an ACP virtual interface, which is placed into the ACP VRF. See Section 6.13.5.2.
8. Each node runs a lightweight routing protocol, see Section 6.12, to announce reachability of the ACP loopback address (or prefix) across the ACP.
9. This completes the creation of the ACP with hop-by-hop secure tunnels, auto-addressing and auto-routing. The node is now an ACP node with a running ACP.

Note:

- * None of the above operations (except the following explicit configured ones) are reflected in the configuration of the node.
- * Non-ACP NMS ("Network Management Systems") or SDN controllers have to be explicitly configured for connection into the ACP.

- * Additional candidate peer adjacencies for ACP connections across non-ACP Layer-3 clouds requires explicit configuration. See Section 8.2.

The following figure illustrates the ACP.

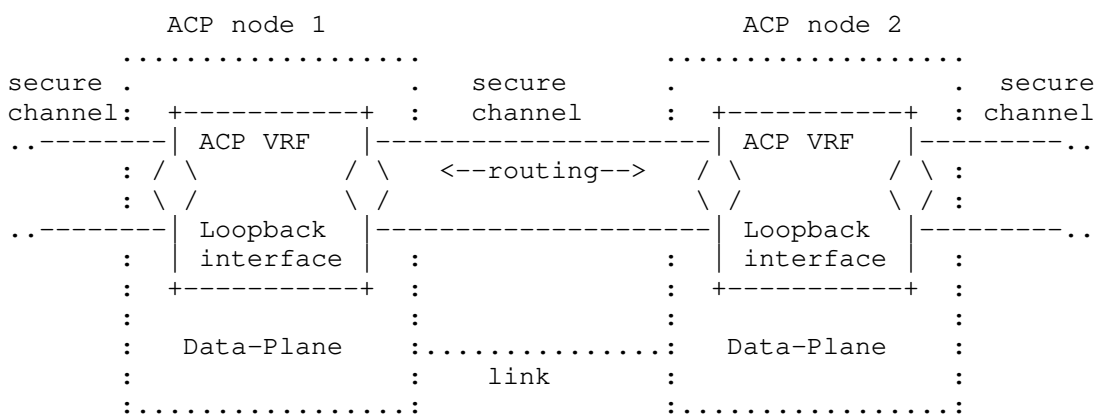


Figure 1: ACP VRF and secure channels

The resulting overlay network is normally based exclusively on hop-by-hop tunnels. This is because addressing used on links is IPv6 link local addressing, which does not require any prior set-up. In this way the ACP can be built even if there is no configuration on the node, or if the Data-Plane has issues such as addressing or routing problems.

6. Self-Creation of an Autonomic Control Plane (ACP) (Normative)

This section specifies the components and steps to set up an ACP. The ACP is automatically "self-creating", which makes it "indestructible" against most changes to the Data-Plane, including misconfigurations of routing, addressing, NAT, firewall or any other traffic policy filters that inadvertently or otherwise unavoidably would also impact the management plane traffic, such as the actual operator CLI session or controller NETCONF session through which the configuration changes to the Data-Plane are executed.

Physical misconfiguration of wiring between ACP nodes will also not break the ACP: As long as there is a transitive physical path between ACP nodes, the ACP should be able to recover given that it automatically operates across all interfaces of the ACP nodes and automatically determines paths between them.

Attacks against the network via incorrect routing or addressing information for the Data-Plane will not impact the ACP. Even impaired ACP nodes will have a significantly reduced attack surface against malicious misconfiguration because only very limited ACP or interface up/down configuration can affect the ACP, and pending on their specific designs these type of attacks could also be eliminated. See more in Section 9.3 and Section 11.

An ACP node can be a router, switch, controller, NMS host, or any other IPv6 capable node. Initially, it MUST have its ACP certificate, as well as an (empty) ACP Adjacency Table (described in Section 6.3). It then can start to discover ACP neighbors and build the ACP. This is described step by step in the following sections:

6.1. Requirements for use of Transport Layer Security (TLS)

The following requirements apply to TLS required or used by ACP components. Applicable ACP components include ACP certificate maintenance via EST, see Section 6.2.5, TLS connections for Certificate Revocation List (CRL) Distribution Point (CRLDP) or Online Certificate Status Protocol (OCSP) responder (if used, see Section 6.2.3) and ACP GRASP (see Section 6.9.2). On ANI nodes these requirements also apply to BRSKI.

TLS MUST comply with [RFC7525] except that TLS 1.2 ([RFC5246]) is REQUIRED and that older versions of TLS MUST NOT be used. TLS 1.3 ([RFC8446]) SHOULD be supported. The choice for TLS 1.2 as the lowest common denominator for the ACP is based on current expected most likely availability across the wide range of candidate ACP node types, potentially with non-agile operating system TCP/IP stacks.

TLS MUST offer TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 and TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 and MUST NOT offer options with less than 256-bit symmetric key strength or hash strength of less than 384 bits. When TLS 1.3 is supported, TLS_AES_256_GCM_SHA384 MUST be offered and TLS_CHACHA20_POLY1305_SHA256 MAY be offered.

TLS MUST also include the "Supported Elliptic Curves" extension, it MUST support the NIST P-256 (secp256r1(22)) and P-384 (secp384r1(24)) curves [RFC8422]. In addition, TLS 1.2 clients SHOULD send an ec_point_format extension with a single element, "uncompressed".

6.2. 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 such as creating ACP secure channels in an autonomous peer-to-peer fashion between ACP domain members via protocols such as IPsec. To authenticate and authorize another ACP member node with access to the ACP Domain, each ACP member requires keying material: An ACP node MUST have a Local Device IDentity (LDevID) certificate, henceforth called the ACP certificate and information about one or more Trust Anchor (TA) as required for the ACP domain membership check (Section 6.2.3).

Manual keying via shared secrets is not usable for an ACP domain because it would require a single shared secret across all current and future ACP domain members to meet the expectation of autonomous, peer-to-peer establishment of ACP secure channels between any ACP domain members. Such a single shared secret would be an unacceptable security weakness. Asymmetric keying material (public keys) without certificates does not provide the mechanisms to authenticate ACP domain membership in an autonomous, peer-to-peer fashion for current and future ACP domain members.

The LDevID certificate is called the ACP certificate. The TA is the Certification Authority (CA) root certificate 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 certificate to comply with Section 6.2.1, specifically to have the acp-node-name as specified in Section 6.2.2 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 except for relying on GRASP and providing security and transport for GRASP. 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

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.

6.2.1. ACP Certificates

ACP certificates MUST be [RFC5280] compliant X.509 v3 ([X.509]) certificates.

ACP nodes MUST support handling ACP certificates, TA certificates and certificate chain certificates (henceforth just called certificates in this section) with RSA public keys and certificates with Elliptic Curve (ECC) public keys.

ACP nodes MUST NOT support certificates with RSA public keys of less than 2048-bit modulus or curves with group order of less than 256-bit. They MUST support certificates with RSA public keys with 2048-bit modulus and MAY support longer RSA keys. They MUST support certificates with ECC public keys using NIST P-256 curves and SHOULD support P-384 and P-521 curves.

ACP nodes MUST NOT support certificates with RSA public keys whose modulus is less than 2048 bits, or certificates whose ECC public keys are in groups whose order is less than 256-bits. RSA signing certificates with 2048-bit public keys MUST be supported, and such certificates with longer public keys MAY be supported. ECDSA certificates using the NIST P-256 curve MUST be supported, and such certificates using the P-384 and P-521 curves SHOULD be supported.

ACP nodes MUST support RSA certificates that are signed by RSA signatures over the SHA-256 digest of the contents, and SHOULD additionally support SHA-384 and SHA-512 digests in such signatures. The same requirements for digest usage in certificate signatures apply to ECDSA certificates, and additionally, ACP nodes MUST support ECDSA signatures on ECDSA certificates.

The ACP certificate SHOULD use an RSA key and an RSA signature when the ACP certificate is intended to be used not only for ACP authentication but also for other purposes. The ACP certificate MAY use an ECC key and an ECDSA signature if the ACP certificate is only used for ACP and ANI authentication and authorization.

Any secure channel protocols used for the ACP as specified in this document or extensions of this document MUST therefore support authentication (e.g. signing) starting with these type of certificates. See [RFC8422] for more information.

The reason for these choices are as follows: As of 2020, RSA is still more widely used than ECC, therefore the MUST for RSA. ECC offers equivalent security at (logarithmically) shorter key lengths (see [RFC8422]). This can be beneficial especially in the presence of constrained bandwidth or constrained nodes in an ACP/ANI network. Some ACP functions such as GRASP peer-2-peer across the ACP require end-to-end/any-to-any authentication/authorization, therefore ECC can only reliably be used in the ACP when it MUST be supported on all ACP nodes. RSA signatures are mandatory to be supported also for ECC certificates because CAs themselves may not support ECC yet.

The ACP certificate SHOULD be used for any authentication between nodes with ACP domain certificates (ACP nodes and NOC nodes) where a required authorization 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 6.2.3 defines this "ACP domain membership check". The uses of this check that are standardized in this document are for the establishment of hop-by-hop ACP secure channels (Section 6.7) and for ACP GRASP (Section 6.9.2) end-to-end via TLS.

The ACP domain membership check requires a minimum amount of elements in a certificate as described in Section 6.2.3. The identity of a node in the ACP is carried via the acp-node-name as defined in Section 6.2.2.

To support ECDH directly with the key in the ACP certificate, ACP certificates with ECC keys need to indicate to be Elliptic Curve Diffie-Hellman capable (ECDH): If the X.509v3 keyUsage extension is present, the keyAgreement bit must then be set. Note that this option is not required for any of the required ciphersuites in this document and may not be supported by all CA.

Any other fields of the ACP certificate are to be populated as required by [RFC5280]: As long as they are compliant with [RFC5280], any other field of an ACP certificate can be set as desired by the operator of the ACP domain through appropriate ACP registrar/ACP CA procedures. For example, other fields may be required for other purposes that the ACP certificate is intended to be used for (such as elements of a SubjectName).

For further certificate details, ACP certificates may follow the recommendations from [CABFORUM].

For diagnostic and other operational purposes, it is beneficial to copy the device identifying fields of the node's IDevID certificate into the ACP certificate, such as the [X.520], section 6.2.9 "serialNumber" attribute in the subject field distinguished name encoding. Note that this is not the certificate serial number. See

also [I-D.ietf-anima-bootstrapping-keyinfra] section 2.3.1. This can be done for example if it would be acceptable for the device's "serialNumber" to be signaled via the Link Layer Discovery Protocol (LLDP, [LLDP]) because like LLDP signaled information, the ACP certificate information can be retrieved by neighboring nodes without further authentication and be used either for beneficial diagnostics or for malicious attacks. Retrieval of the ACP certificate is possible via a (failing) attempt to set up an ACP secure channel, and the "serialNumber" usually contains device type information that may help to faster determine working exploits/attacks against the device.

Note that there is no intention to constrain authorization within the ACP or autonomic networks using the ACP to just the ACP domain membership check as defined in this document. It can be extended or modified with additional requirements. Such future authorizations can use and require additional elements in certificates or policies or even additional certificates. See the additional check against the id-kp-cmcRA [RFC6402] extended key usage attribute (Section 6.2.5) and for possible future extensions, see Appendix A.9.5.

6.2.2. ACP Certificate AcpNodeName

```

acp-node-name = local-part "@" acp-domain-name
local-part = [ acp-address ] [ "+" rsub extensions ]
acp-address = 32HEXDIG | "0" ; HEXDIG as of RFC5234 section B.1
rsub = [ <subdomain> ] ; <subdomain> as of RFC1034, section 3.5
acp-domain-name = ; <domain> ; as of RFC 1034, section 3.5
extensions = *( "+" extension )
extension = 1*etext ; future standard definition.
etext      = ALPHA / DIGIT / ; Printable US-ASCII
            "!" / "#" / "$" / "%" / "&" / "'" /
            "*" / "-" / "/" / "=" / "?" / "^" /
            "_" / "`" / "{" / "|" / "}" / "~"

routing-subdomain = [ rsub "." ] acp-domain-name

```

Example:

```

given an ACP address   of fd89:b714:f3db:0:200:0:6400:0000
and an ACP domain-name of acp.example.com
and an rsub extension of area51.research

```

then this results in:

```

acp-node-name      = fd89b714f3db00000200000064000000
                   +area51.research@acp.example.com
acp-domain-name    = acp.example.com
routing-subdomain  = area51.research.acp.example.com

```

Figure 2: ACP Node Name ABNF

acp-node-name in above Figure 2 is the ABNF ([RFC5234]) definition of the ACP Node Name. An ACP certificate MUST carry this information. It MUST be encoded as a subjectAltName / otherName / AcpNodeName as described in Section 6.2.2.1.

Nodes complying with this specification MUST be able to receive their ACP address through the domain certificate, in which case their own ACP certificate MUST have a 32HEXDIG acp-address field. Acp-address is case insensitive because ABNF HEXDIG is. It is recommended to encode acp-address with lower case letters. Nodes complying with this specification MUST also be able to authenticate nodes as ACP domain members or ACP secure channel peers when they have a 0-value acp-address field and as ACP domain members (but not as ACP secure channel peers) when the acp-address field is omitted from their AcpNodeName. See Section 6.2.3.

acp-domain-name is used to indicate the ACP Domain across which ACP nodes authenticate and authorize each other, for example to build ACP secure channels to each other, see Section 6.2.3. acp-domain-name SHOULD be the FQDN of an Internet domain owned by the network administration of the ACP and ideally reserved to only be used for the ACP. In this specification it serves to be a name for the ACP that ideally is globally unique. When acp-domain-name is a globally unique name, collision of ACP addresses across different ACP domains can only happen due to ULA hash collisions (see Section 6.11.2). Using different acp-domain-names, operators can distinguish multiple ACP even when using the same TA.

To keep the encoding simple, there is no consideration for internationalized acp-domain-names. The acp-node-name is not intended for end user consumption. There is no protection against an operator to pick any domain name for an ACP whether or not the operator can claim to own the domain name. Instead, the domain name only serves as a hash seed for the ULA and for diagnostics to the operator. Therefore, any operator owning only an internationalized domain name should be able to pick an equivalently unique 7-bit ASCII acp-domain-name string representing the internationalized domain name.

"routing-subdomain" is a string that can be constructed from the acp-node-name, and it is used in the hash-creation of the ULA (see below). The presence of the "rsub" component allows a single ACP domain to employ multiple /48 ULA prefixes. See Appendix A.6 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.

The following points explain and justify the encoding choices described:

1. Formatting notes:
 - 1.1 "rsub" needs to be in the "local-part": If the format just had routing-subdomain as the domain part of the acp-node-name, rsub and acp-domain-name could not be separated from each other to determine in the ACP domain membership check which part is the acp-domain-name and which is solely for creating a different ULA prefix.
 - 1.2 If both "acp-address" and "rsub" are omitted from AcpNodeName, the "local-part" will have the format "++extension(s)". The two plus characters are necessary so the node can unambiguously parse that both "acp-address" and "rsub" are omitted.
2. The encoding of the ACP domain name and ACP address as described in this section is used for the following reasons:
 - 2.1 The acp-node-name is the identifier of a node's ACP. It includes the necessary components to identify a node's ACP both from within the ACP as well as from the outside of the ACP.
 - 2.2 For manual and/or automated diagnostics and backend management of devices and ACPs, it is necessary to have an easily human readable and software parsed standard, single string representation of the information in the acp-node-name. For example, inventory or other backend systems can always identify an entity by one unique string field but not by a combination of multiple fields, which would be necessary if there was no single string representation.
 - 2.3 If the encoding was not that of such a string, it would be necessary to define a second standard encoding to provide this format (standard string encoding) for operator consumption.
 - 2.4 Addresses of the form <local>@<domain> have become the preferred format for identifiers of entities in many systems, including the majority of user identification in web or mobile applications such as multi-domain single-sign-on systems.
3. Compatibilities:
 - 3.1 It should be possible to use the ACP certificate as an LDevID certificate on the system for 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

attributes of the subject field for example are often used in non-ACP applications and should therefore not be occupied by new ACP values.

- 3.2 The element should not require additional ASN.1 en/decoding, because libraries to access certificate information especially for embedded devices may not support extended ASN.1 decoding beyond predefined, mandatory fields. subjectAltName / otherName is already used with a single string parameter for several otherNames (see [RFC3920], [RFC7585], [RFC4985], [RFC8398]).
- 3.3 The element required for the ACP should minimize the risk of being misinterpreted by other uses of the LDevID certificate. It also must not be misinterpreted to actually be an email address, hence the use of the otherName / rfc822Name option in the certificate would be inappropriate.

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

6.2.2.1. AcpNodeName ASN.1 Module

The following ASN.1 module normatively specifies the AcpNodeName structure. This specification uses the ASN.1 definitions from [RFC5912] with the 2002 ASN.1 notation used in that document. [RFC5912] updates normative documents using older ASN.1 notation.

```

ANIMA-ACP-2020
{ iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
  id-mod-anima-acpnode-name-2020(IANA1) }

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS
  OTHER-NAME
  FROM PKIX1Implicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-implicit-02(59) }

  id-pkix
  FROM PKIX1Explicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51) } ;

  id-on OBJECT IDENTIFIER ::= { id-pkix 8 }

  AcpNodeNameOtherNames OTHER-NAME ::= { on-AcpNodeName, ... }

  on-AcpNodeName OTHER-NAME ::= {
    AcpNodeName IDENTIFIED BY id-on-AcpNodeName
  }

  id-on-AcpNodeName OBJECT IDENTIFIER ::= { id-on IANA2 }

  AcpNodeName ::= IA5String (SIZE (1..MAX))
    -- AcpNodeName as specified in this document carries the
    -- acp-node-name as specified in the ABNF in Section 6.1.2

END

```

Figure 3

6.2.3. ACP domain membership check

The following points constitute the ACP domain membership check of a candidate peer via its certificate:

- 1: The peer has proved ownership of the private key associated with the certificate's public key. This check is performed by the security association protocol used, for example [RFC7296], section 2.15.

- 2: The peer's certificate passes certificate path validation as defined in [RFC5280], section 6 against one of the TA associated with the ACP node's ACP certificate (see Section 6.2.4 below). This includes verification of the validity (lifetime) of the certificates in the path.
- 3: If the peer's certificate indicates a Certificate Revocation List (CRL) Distribution Point (CRLDP) ([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 mechanisms when they are available: 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 CRLDP. These mechanisms are not available when the ACP node has no ACP or non-ACP connectivity to retrieve a current CRL or access an OCSP responder and the security association protocol itself has also no way to communicate CRL or OCSP check. Retries to learn revocation via OCSP/CRL SHOULD be made using the same backoff as described in Section 6.7. If and when the ACP node then learns that an ACP peer's certificate is invalid for which rule 3 had to be skipped during ACP secure channel establishment, then the ACP secure channel to that peer MUST be closed even if this peer is the only connectivity to access CRL/OCSP. This applies (of course) to all ACP secure channels to this peer if there are multiple. The ACP secure channel connection MUST be retried periodically to support the case that the neighbor acquires a new, valid certificate.
- 4: The peer's certificate has a syntactically valid acp-node-name field and the acp-domain-name in that peer's acp-node-name is the same as in this ACP node's certificate (lowercase normalized).

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

- 5: The acp-address field of the candidate peer certificate's AcpNodeName is not omitted but either 32HEXDIG or 0, according to Figure 2.

Technically, ACP secure channels can only be built with nodes that have an acp-address. Rule 5 ensures that this is taken into account during ACP domain membership check.

Nodes with an omitted acp-address field can only use their ACP domain certificate for non-ACP-secure channel authentication purposes. This includes for example NMS type nodes permitted to communicate into the ACP via ACP connect (Section 8.1)

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 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, but this option is defined here so that any ACP nodes can build ACP secure channels to them according to Rule 5.

The optional rsub field of the AcpNodeName is not relevant to the ACP domain membership check because it only serves to structure routing and addressing within an ACP but not to segment mutual authentication/authorization (hence the name "routing subdomain").

In summary:

- * Steps 1...4 constitute standard certificate validity verification and private key authentication as defined by [RFC5280] and security association protocols (such as Internet Key Exchange Protocol version 2 IKEv2 [RFC7296] when leveraging certificates.
- * Steps 1...4 do not include verification of any pre-existing form of non-public-key-only based identity elements of a certificate such as a web servers domain name prefix often encoded in certificate common name. Step 5 is an equivalent step for the AcpNodeName.
- * Step 4 constitutes standard CRL/OCSP checks refined for the case of missing connectivity and limited functionality security association protocols.
- * Steps 1...4 authorize to build any secure connection between members of the same ACP domain except for ACP secure channels.
- * Step 5 is the additional verification of the presence of an ACP address as necessary for ACP secure channels.
- * Steps 1...5 therefore authorize 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.

[RFC-Editor: Please remove the following paragraph].

Note that the ACP domain membership check does not verify the network layer address of the security association. See [ACPDRAFT], Appendix B.2 for explanations.

6.2.3.1. Realtime clock and Time Validation

An ACP node with a realtime clock in which it has confidence, **MUST** check the time stamps when performing ACP domain membership check such as the certificate validity period in step 1. and the respective times in step 4 for revocation information (e.g., signingTimes in CMS signatures).

An ACP node without such a realtime clock **MAY** ignore those time stamp validation steps if it does not know the current time. Such an ACP node **SHOULD** obtain the current time in a secured fashion, such as via a Network Time Protocol signaled through the ACP. It then ignores time stamp validation only until the current time is known. In the absence of implementing a secured mechanism, such an ACP node **MAY** use a current time learned in an insecure fashion in the ACP domain membership check.

Current time **MAY** for example be learned unsecured via NTP ([RFC5905]) over the same link-local IPv6 addresses used for the ACP from neighboring ACP nodes. ACP nodes that do provide NTP insecure over their link-local addresses **SHOULD** primarily run NTP across the ACP and provide NTP time across the ACP only when they have a trusted time source. Details for such NTP procedures are beyond the scope of this specification.

Beside ACP domain membership check, the ACP itself has no dependency against knowledge of the current time, but protocols and services using the ACP will likely have the need to know the current time. For example, event logging.

6.2.4. Trust Anchors (TA)

ACP nodes need TA information according to [RFC5280], section 6.1.1 (d), typically in the form of one or more certificate of the TA to perform certificate path validation as required by Section 6.2.3, rule 2. TA information **MUST** be provisioned to an ACP node (together with its ACP domain certificate) by an ACP Registrar during initial enrollment of a candidate ACP node. ACP nodes **MUST** also support renewal of TA information via EST as described below in Section 6.2.5.

The required information about a TA can consist of not only a single, but multiple certificates as required for dealing with CA certificate renewals as explained in Section 4.4 of CMP ([RFC4210]).

A certificate path is a chain of certificates starting at the ACP certificate (leaf/end-entity) followed by zero or more intermediate CA certificates and ending with the TA information, which are

typically one or two the self-signed certificates of the TA. The CA that signs the ACP certificate is called the assigning CA. If there are no intermediate CA, then the assigning CA is the TA. Certificate path validation authenticates that the ACP certificate is permitted by a TA associated with the ACP, directly or indirectly via one or more intermediate CA.

Note that different ACP nodes may have different intermediate CA in their certificate path and even different TA. The set of TA for an ACP domain must be consistent across all ACP members so that any ACP node can authenticate any other ACP node. The protocols through which ACP domain membership check rules 1-3 are performed need to support the exchange not only of the ACP nodes certificates, but also exchange of the intermediate TA.

ACP nodes MUST support for the ACP domain membership check the certificate path validation with 0 or 1 intermediate CA. They SHOULD support 2 intermediate CA and two TA (to permit migration to from one TA to another TA).

Certificates for an ACP MUST only be given to nodes that are allowed to be members of that ACP. When the signing CA relies on an ACP Registrar, the CA MUST only sign certificates with acp-node-name through trusted ACP Registrars. In this setup, any existing CA, unaware of the formatting of acp-node-name, can be used.

These requirements can be achieved by using a TA private to the owner of the ACP domain or potentially through appropriate contractual agreements between the involved parties (Registrar and CA). Using public CA is out of scope of this document. [RFC-Editor: please remove the following sentence]. See [ACPDRAFT], Appendix B.3 for further considerations.

A single owner can operate multiple independent ACP domains from the same set of TA. Registrars must then know which ACP a node needs to be enrolled into.

6.2.5. Certificate and Trust Anchor Maintenance

ACP nodes MUST support renewal of their Certificate and TA information via EST and MAY support other mechanisms. See Section 6.1 for TLS requirements. 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 IPv6 locator parameters of the O_IPv6_LOCATOR in GRASP of the EST server from which they last renewed their ACP certificate. They SHOULD provide the ability for

these EST server parameters to also be set by the ACP Registrar (see Section 6.11.7) that initially enrolled the ACP device with its ACP certificate. When BRSKI (see [I-D.ietf-anima-bootstrapping-keyinfra]) is used, the IPv6 locator 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.

The EST server MUST present a certificate that is passing ACP domain membership check in its TLS connection setup (Section 6.2.3, rules 1...4, not rule 5 as this is not for an ACP secure channel setup). The EST server certificate MUST also contain the id-kp-cmcRA [RFC6402] extended key usage attribute and the EST client MUST check its presence.

The additional check against the id-kp-cmcRA extended key usage extension field ensures that clients do not fall prey to an illicit EST server. While such illicit EST servers should not be able to support certificate signing requests (as they are not able to elicit a signing response from a valid CA), such an illicit EST server would be able to provide faked CA certificates to EST clients that need to renew their CA certificates when they expire.

Note that EST servers supporting multiple ACP domains will need to have for each of these ACP domains a separate certificate and respond on a different transport address (IPv6 address and/or TCP port), but this is easily automated on the EST server as long as the CA does not restrict registrars to request certificates with the id-kp-cmcRA extended usage extension for themselves.

6.2.5.1. 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'fd89b714f3db0000200000064000001', 210000,
  [{"SRV.est", 4, 255 },
   [O_IPv6_LOCATOR,
    h'fd89b714f3db0000200000064000001', IPPROTO_TCP, 443]]
]
```

Figure 4: GRASP SRV.est example

The formal definition of the objective in Concise data definition language (CDDL) (see [RFC8610]) is as follows:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                  +[objective, (locator-option / [])]]
                  ; see example above and explanation
                  ; below for initiator and ttl

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 as there is no mechanism
                           ; to discover network diameter.
objective-value = any       ; reserved for future extensions
```

Figure 5: 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. Non-recognized objective-values (or parts thereof if it is a structure partially understood) MUST be ignored.

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 instance is dead and select another instance of the same service instead (from another GRASP announcement).

The "SRV.est" objective(s) SHOULD only be announced when the ACP node knows that it can successfully communicate with a CA to perform the EST renewal/rekeying operations for the ACP domain. See also Section 11.

6.2.5.2. 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.

6.2.5.3. Certificate Revocation Lists (CRLs)

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

It is common to use domain names for CRLDP(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 CRLDP via the ACP.

A CRLDP 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).

When using a private PKI for ACP certificates, the CRL may be need-to-know, for example to prohibit insight into the operational practices of the domain by tracking the growth of the CRL. In this case, HTTPS may be chosen to provide confidentiality, especially when making the CRL available via the Data-Plane. Authentication and authorization SHOULD use ACP certificates and ACP domain membership check. The CRLDP MAY omit the CRL verification during authentication of the peer to permit retrieval of the CRL by an ACP node with revoked ACP certificate. This can allow for that (ex) ACP node to quickly discover its ACP certificate revocation. This may violate the desired need-to-know requirement though. ACP nodes MAY support CRLDP operations via HTTPS.

6.2.5.4. 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 certificates are managed via a CA chain where the assigning CA has enough performance to manage short lived certificates. See also Section 9.2.4 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]).

6.2.5.5. Re-enrollment

An ACP node may determine that its ACP 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 TA and certificate chain associated with its ACP 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 6.11.7 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 certificate, 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 node's acp-node-name, so that the BRSKI registrar can re-assign the same ACP address information to the ACP node in the new ACP certificate.

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

Both when the BRSKI connection is attempted with the old ACP certificate or the IDevID certificate, the re-enrolling candidate ACP node SHOULD authenticate the BRSKI registrar during TLS connection setup based on its existing TA certificate chain information associated with its old ACP certificate. The re-enrolling candidate ACP node SHOULD only fall back to requesting a voucher from the BRSKI registrar when this authentication fails during TLS connection setup. As a countermeasure against attacks that attempt to force the ACP node to forget its prior (expired) certificate and TA, the ACP node should alternate between attempting to re-enroll using its old keying material and attempting to re-enroll with its IDevID and requesting a voucher.

When other mechanisms than BRSKI are used for ACP 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/TA information and provides its existing ACP certificate and other identification (such as the IDevID certificate) as necessary to the registrar.

Maintaining existing TA information is especially important when enrollment mechanisms are used that unlike BRSKI do not leverage a mechanism (such as the voucher in BRSKI) to authenticate the ACP registrar and where therefore the injection of certificate failures could otherwise make the ACP node easily attackable remotely by

returning the ACP node to a "duckling" state in which it accepts to be enrolled by any network it connects to. The (expired) ACP certificate and ACP TA SHOULD therefore be maintained and attempted to be used as one possible credential for re-enrollment until new keying material is acquired.

When using BRSKI or other protocol/mechanisms supporting vouchers, maintaining existing TA 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.

6.2.5.6. Failing Certificates

An ACP certificate is called failing in this document, if/when the ACP node to which the certificate was issued 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 certificate as the culprit, the peer should pass the domain membership check (Section 6.2.3) 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 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 6.2.5.5).

6.3. 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 6.11.3 and Section 6.11.5), interface on which neighbor was discovered (by GRASP as explained below), link-local IPv6 address of neighbor on that interface, certificate (including acp-node-name). 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 6.2.3).

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.

6.4. 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 ([RFC4541]) switches that are not ACP supporting/enabled - because those switches 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 9.3 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 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', IPPROTO_UDP, 15000]]
 [{"AN_ACP", 4, 1, "DTLS" },
  [O_IPv6_LOCATOR,
   h'fe80000000000000c0011001feef0000', IPPROTO_UDP, 17000]]
]
```

Figure 6: 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-name / [ method, *extension ]
method = method-name / [ method-name, *method-param ]
method-name = "IKEv2" / "DTLS" / id
extension = any
method-param = any
id = text .regexp "[A-Za-z@_$(-)]*[A-Za-z0-9@_$(-)]*"

```

Figure 7: 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 has to 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 method-name in the 'objective-value' parameter is a string indicating the protocol available at the specified or implied locator. It is a protocol supported by the node to negotiate a secure channel. IKEv2 as shown above is the protocol used to negotiate an IPsec secure channel.

Method-params allows to carry method specific parameters. This specification does not define any method-param(s) for "IKEv2" or "DTLS". Method-params for these two methods that are not understood by an ACP node MUST be ignored by it.

extension(s) allows to define method independent parameters. This specification does not define any extensions. Extensions not understood by an ACP node MUST be ignored by it.

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 actual protocol used to negotiate an Internet Protocol security architecture (IPsec) connection. GRASP therefore indicates "IKEv2" and not "IPsec". If "IPsec" was used, this too could mean use of the obsolete older version IKE (v1) ([RFC2409]). IKEv2 has an IANA assigned port number 500, but in the above example, the candidate ACP neighbor is offering ACP secure channel negotiation via IKEv2 on port 15000 (purely to show through the example that GRASP allows to indicate the port number and it does not have to be the IANA assigned one).

There is no default UDP port for DTLS, it is always locally assigned by the node. For further details about the "DTLS" secure channel protocol, see Section 6.8.4.

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 6.

The use of DULL GRASP primarily serves to discover the link-local IPv6 address of candidate ACP peers on subnets. The signaling of the supported secure channel option is primarily for diagnostic purposes, but it is also necessary for discovery when the protocol has no well-known transport address, such as in the case of DTLS. [RFC-Editor: Please remove the following sentence]. See [ACPDRAFT], Appendix B.4.

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 6.3), which then drives the further building of the ACP to that neighbor.

Note that the DULL GRASP objective described intentionally does not include the ACP node's ACP certificate even though this would be useful for diagnostics and to simplify the security exchange in ACP secure channel security association protocols (see Section 6.8). The reason is that DULL GRASP messages are periodically multicasted across IPv6 subnets and full certificates could easily lead to fragmented IPv6 DULL GRASP multicast packets due to the size of a certificate. This would be highly undesirable.

6.5. Candidate ACP Neighbor Selection

An ACP node determines to which other ACP nodes in the adjacency table it should attempt to 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.6 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.

6.6. 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.

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. Instead, MacSec is mentioned as a likely next interesting secure channel protocol. Note also that the security model allows and requires for any-to-any authentication and authorization between all ACP nodes because there is also end-to-end and not only hop-by-hop authentication for secure channels.

To support extensible secure channel protocol selection without a single common mandatory to implement (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 ACP secure channel protocol connection to a specific peer IPv6 address passes peer authentication, the two peers know each other's certificate because those ACP certificates are used by all secure channel protocols for mutual authentication. The peer with the higher Node-ID in the AcpNodeName of its ACP certificate takes on the role of the Decider towards the peer. The other peer takes on the role of the Follower. The Decider selects which secure channel protocol to ultimately use.
- * The Follower becomes passive: it does not attempt to further initiate ACP secure channel protocol connections with the Decider and does not consider it to be an error when the Decider closes secure channels. The Decider becomes the active party, continues to attempt setting up secure channel protocols with the Follower. This process terminates when the Decider arrives at the "best" ACP secure channel connection option that also works with the Follower ("best" from the Deciders point of view).
- * A peer with a "0" acp-address in its AcpNodeName takes on the role of Follower when peering with a node that has a non-"0" acp-address (note that this specification does not fully define the

behavior of ACP secure channel negotiation for nodes with a "0" ACP address field, it only defines interoperability with such ACP nodes).

In a simple example, ACP peer Node 1 attempts to initiate an IPsec via IKEv2 connection to peer Node 2. The IKEv2 authentication succeeds. Node 1 has the lower ACP address and becomes the Follower. Node 2 becomes the Decider. IKEv2 might not be the preferred ACP secure channel protocol for the Decider Node 2. Node 2 would therefore proceed to attempt secure channel setups with (in its view) more preferred protocol options (e.g., DTLS/UDP). If any such preferred ACP secure channel connection of the Decider succeeds, it would close the IPsec connection. If Node 2 has no preferred protocol option over IPsec, or no such connection attempt from Node 2 to Node 1 succeeds, Node 2 would keep the IPsec connection and use it.

The Decider SHOULD NOT send actual payload packets across a secure channel until it has decided to use it. The Follower MAY delay linking the ACP secure channel into the ACP virtual interface until it sees the first payload packet from the Decider up to a maximum of 5 seconds to avoid unnecessarily linking a secure channel that will be terminated as undesired by the Decider shortly afterwards.

The following sequence of steps show this example in more detail. Each step is tagged with [<step#>{:<connection>}]. The connection is included to more easily distinguish which of the two competing connections the step belongs to, one initiated by Node 1, one initiated by Node 2.

- [1] Node 1 sends GRASP AN_ACP message to announce itself
- [2] Node 2 sends GRASP AN_ACP message to announce itself
- [3] Node 2 receives [1] from Node 1
- [4:C1] Because of [3], Node 2 starts as initiator on its preferred secure channel protocol towards Node 1. Connection C1.
- [5] Node 1 receives [2] from Node 2
- [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 the Follower and Node 2 the Decider on connection C1. Connection setup C1 is completed.
- [9] Node 1 refrains from attempting any further secure channel connections to Node 2 (the Decider) as learned from [2] because it knows from [8:C1] that it is the Follower relative to Node 1.
- [10:C2] Node1 and Node2 have authenticated each others certificate on connection C2 (like [7:C1]).
- [11:C2] Node 1 certificate has lower ACP Node-ID than Node2, therefore Node 1 considers itself the Follower and Node 2 the Decider on connection C2, but they also identify that C2 is to the same mutual peer as their C1, so this has no further impact: the roles Decider and Follower where already assigned between these two peers by [8:C1].
- [12:C2] Node 2 (the Decider) closes C1. Node 1 is fine with this, because of its role as the Follower (from [8:C1]).
- [13] Node 2 (the Decider) and Node 1 (the Follower) start data transfer across C2, which makes it become a secure channel for the ACP.

Figure 8: Secure Channel sequence of steps

All this negotiation is in the context of an "L2 interface". The Decider and Follower will build ACP connections to each other on every "L2 interface" that they both connect to. 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.

The Decider SHOULD NOT suppress attempting a particular ACP secure channel protocol connection on one L2 interface because this type of ACP secure channel connection has failed to the peer with the same ACP certificate on another L2 interface: Not only the supported ACP secure channel protocol options may be different on the same ACP peer across different L2 interfaces, but also error conditions may cause inconsistent failures across different L2 interfaces. Avoiding such connection attempt optimizations can therefore help to increase robustness in the case of errors.

6.7. 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 6.2.3. 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 an initial retransmission time (IRT) of 10 seconds and a maximum retransmission time (MRT) of 640 seconds.

Failure to authenticate an ACP neighbor when acting in the role of a responder of the security authentication protocol MUST NOT impact the attempts of the ACP node to attempt establishing a connection as an initiator. Only failed connection attempts as an initiator must cause throttling. This rule is meant to increase resilience of secure channel creation. Section 6.6 shows how simultaneous mutual secure channel setup collisions are resolved.

6.8. Security Association (Secure Channel) protocols

This section describes how ACP nodes establish secured data connections to automatically discovered or configured peers in the ACP. Section 6.4 above described how IPv6 subnet adjacent peers are discovered automatically. Section 8.2 describes how non IPv6 subnet adjacent peers can be configured.

Section 6.13.5.2 describes how secure channels are mapped to virtual IPv6 subnet interfaces in the ACP. The simple case is to map every ACP secure channel into a separate ACP point-to-point virtual interface Section 6.13.5.2.1. When a single subnet has multiple ACP peers this results in multiple ACP point-to-point virtual interfaces across that underlying multi-party IPv6 subnet. This can be optimized with ACP multi-access virtual interfaces (Section 6.13.5.2.2) but the benefits of that optimization may not justify the complexity of that option.

6.8.1. General considerations

Due to Channel Selection (Section 6.6), ACP can support an evolving set of security association protocols and does not require support for a single network wide MTI. ACP nodes only need to implement those protocols required to interoperate with their candidate peers, not with potentially any node in the ACP domain. See Section 6.8.5 for an example of this.

The degree of security required on every hop of an ACP network needs to be consistent across the network so that there is no designated "weakest link" because it is that "weakest link" that would otherwise become the designated point of attack. When the secure channel protection on one link is compromised, it can be used to send/receive packets across the whole ACP network. Therefore, even though the security association protocols can be different, their minimum degree of security should be comparable.

Secure channel protocols do not need to always support arbitrary L3 connectivity between peers, but can leverage the fact that the standard use case for ACP secure channels is an L2 adjacency. Hence, L2 dependent mechanisms could be adopted for use as secure channel association protocols:

L2 mechanisms such as strong encrypted radio technologies or [MACSEC] may offer equivalent encryption and the ACP security association protocol may only be required to authenticate ACP domain membership of a peer and/or derive a key for the L2 mechanism. Mechanisms to auto-discover and associate ACP peers leveraging such underlying L2 security are possible and desirable to avoid duplication of encryption, but none are specified in this document.

Strong physical security of a link may stand in where cryptographic security is infeasible. As there is no secure mechanism to automatically discover strong physical security solely between two peers, it can only be used with explicit configuration and that configuration too could become an attack vector. This document therefore only specifies with ACP connect (Section 8.1) one

explicitly configured mechanism without any secure channel association protocol – for the case where both the link and the nodes attached to it have strong physical security.

6.8.2. Common requirements

The authentication of peers in any security association protocol MUST use the ACP certificate according to Section 6.2.3. Because auto-discovery of candidate ACP neighbors via GRASP (see Section 6.4) as specified in this document does not communicate the neighbors ACP certificate, and ACP nodes may not (yet) have any other network connectivity to retrieve certificates, any security association protocol MUST use a mechanism to communicate the certificate directly instead of relying on a referential mechanism such as communicating only a hash and/or URL for the certificate.

A security association protocol MUST use Forward Secrecy (whether inherently or as part of a profile of the security association protocol).

Because the ACP payload of legacy protocol payloads inside the ACP and hop-by-hop ACP flooded GRASP information is unencrypted, the ACP secure channel protocol requires confidentiality. Symmetric encryption for the transmission of secure channel data MUST use encryption schemes considered to be security wise equal to or better than 256-bit key strength, such as AES256. There MUST NOT be support for NULL encryption.

Security association protocols typically only signal the End Entity certificate (e.g. the ACP certificate) and any possible intermediate CA certificates for successful mutual authentication. The TA has to be mutually known and trusted and therefore its certificate does not need to be signaled for successful mutual authentication. Nevertheless, for use with ACP secure channel setup, there SHOULD be the option to include the TA certificate in the signaling to aid troubleshooting, see Section 9.1.1.

Signaling of TA certificates may not be appropriate when the deployment is relying on a security model where the TA certificate content is considered confidential and only its hash is appropriate for signaling. ACP nodes SHOULD have a mechanism to select whether the TA certificate is signaled or not. Assuming that both options are possible with a specific secure channel protocol.

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. This may not be standard behavior in secure channel protocols because the certificate

authentication may only influence the setup of the secure channel in these protocols, but may not be re-validated during the lifetime of the secure connection in the absence of this requirement.

When specifying an additional security association protocol for ACP secure channels beyond those covered in this document, protocol options SHOULD be eliminated that are not necessary to support devices that are expected to be able to support the ACP to minimize implementation complexity. For example, definitions for security protocols often include old/inferior security options required only to interoperate with existing devices that will not be able to update to the currently preferred security options. Such old/inferior security options do not need to be supported when a security association protocol is first specified for the ACP, strengthening the "weakest link" and simplifying ACP implementation overhead.

6.8.3. ACP via IPsec

An ACP node announces its ability to support IPsec, negotiated via IKEv2, as the ACP secure channel protocol using the "IKEv2" objective-value in the "AN_ACP" GRASP objective.

The ACP usage of IPsec and IKEv2 mandates a profile with a narrow set of options of the current standards-track usage guidance for IPsec [RFC8221] and IKEv2 [RFC8247]. These options result in stringent security properties and can exclude deprecated/legacy algorithms because there is no need for interoperability with legacy equipment for ACP secure channels. Any such backward compatibility would lead only to increased attack surface and implementation complexity, for no benefit.

6.8.3.1. Native IPsec

An ACP node that is supporting native IPsec MUST use IPsec in tunnel mode, negotiated via IKEv2, and with IPv6 payload (e.g., ESP Next Header of 41). It MUST use local and peer link-local IPv6 addresses for encapsulation. Manual keying MUST NOT be used, see Section 6.2. Traffic Selectors are:

TSi = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)

TSr = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)

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 (and no additional encapsulation header in the ESP payload), it would only be possible to send packets originated by the ACP node itself because the IPv6 addresses of the ESP must be the same as that of the outer IPv6 header.

6.8.3.1.1. RFC8221 (IPsec/ESP)

ACP IPsec implementations MUST comply with [RFC8221] (and its updates). The requirements from above and this section amend and superseded its requirements.

The IP Authentication Header (AH) MUST NOT be used (because it does not provide confidentiality).

For the required ESP encryption algorithms in section 5 of [RFC8221] the following guidance applies:

- * ENCR_NULL AH MUST NOT be used (because it does not provide confidentiality).
- * ENCR_AES_GCM_16 is the only MTI ESP encryption algorithm for ACP via IPsec/ESP (it is already listed as MUST in [RFC8221]).
- * ENCR_AES_CBC with AUTH_HMAC_SHA2_256_128 (as the ESP authentication algorithm) and ENCR_AES_CCM_8 MAY be supported. If either provides higher performance than ENCR_AES_GCM_16 it SHOULD be supported.
- * ENCR_CHACHA20_POLY1305 SHOULD be supported at equal or higher performance than ENCR_AES_GCM_16. If that performance is not feasible, it MAY be supported.

IKEv2 indicates an order for the offered algorithms. The algorithms SHOULD be ordered by performance. The first algorithm supported by both sides is generally chosen.

Explanations:

- * There is no requirement to interoperate with legacy equipment in ACP secure channels, so a single MTI encryption algorithm for IPsec in ACP secure channels is sufficient for interoperability and allows for the most lightweight implementations.
- * ENCR_AES_GCM_16 is an authenticated encryption with associated data (AEAD) cipher mode, so no additional ESP authentication algorithm is needed, simplifying the MTI requirements of IPsec for the ACP.

- * There is no MTI requirement for the support of ENCR_AES_CBC because ENCR_AES_GCM_16 is assumed to be feasible with less cost/higher performance in modern devices hardware accelerated implementations compared to ENCR_AES_CBC.
- * ENCR_CHACHA20_POLY1305 is mandatory in [RFC8221] because of its target use as a fallback algorithm in case weaknesses in AES are uncovered. Unfortunately, there is currently no way to automatically propagate across an ACP a policy to disallow use of AES based algorithms, so this target benefit of ENCR_CHACHA20_POLY1305 cannot fully be adopted yet for the ACP. Therefore, this algorithm is only recommended. Changing from AES to this algorithm at potentially big drop in performance could also render the ACP inoperable. Therefore, the performance requirement against this algorithm so that it could become an effective security backup to AES for the ACP once a policy to switch over to it or prefer it is available in an ACP framework.

[RFC8221] allows for 128-bit or 256-bit AES keys. This document mandates that only 256-bit AES keys MUST be supported.

When [RFC8221] is updated, ACP implementations will need to consider legacy interoperability, and the IPsec WG has generally done a very good job of taking that into account in its recommendations.

6.8.3.1.2. RFC8247 (IKEv2)

[RFC8247] provides a baseline recommendation for mandatory to implement ciphers, integrity checks, pseudo-random-functions and Diffie-Hellman mechanisms. Those recommendations, and the recommendations of subsequent documents apply well to the ACP. Because IKEv2 for ACP secure channels is sufficient to be implemented in control plane software, rather than in ASIC hardware, and ACP nodes supporting IKEv2 are not assumed to be code-space constrained, and because existing IKEv2 implementations are expected to support [RFC8247] recommendations, this documents makes no attempt to simplify its recommendations for use with the ACP.

See [IKEV2IANA] for IANA IKEv2 parameter names used in this text.

ACP Nodes supporting IKEv2 MUST comply with [RFC8247] amended by the following requirements which constitute a policy statement as permitted by [RFC8247].

To signal the ACP certificate chain (including TA) as required by Section 6.8.2, "X.509 Certificate - Signature" payload in IKEv2 can be used. It is mandatory according to [RFC7296] section 3.6.

ACP nodes SHOULD set up IKEv2 to only use the ACP certificate and TA when acting as an IKEv2 responder on the IPv6 link local address and port number indicated in the AN_ACP DULL GRASP announcements (see Section 6.4).

When CERTREQ is received from a peer, and does not indicate any of this ACP nodes TA certificates, the ACP node SHOULD ignore the CERTREQ and continue sending its certificate chain including its TA as subject to the requirements and explanations in Section 6.8.2. This will not result in successful mutual authentication but assists diagnostics.

Note that with IKEv2, failing authentication will only result in the responder receiving the certificate chain from the initiator, but not vice versa. Because ACP secure channel setup is symmetric (see Section 6.7), every non-malicious ACP neighbor will attempt to connect as an initiator though, allowing to obtain the diagnostic information about the neighbors certificate.

In IKEv2, ACP nodes are identified by their ACP address. The ID_IPv6_ADDR IKEv2 identification payload MUST be used and MUST convey the ACP address. If the peer's ACP certificate includes a 32HEXDIG ACP address in the acp-node-name (not "0" or omitted), the address in the IKEv2 identification payload MUST match it. See Section 6.2.3 for more information about "0" or omitted ACP address fields in the acp-node-name.

IKEv2 authentication MUST use authentication method 14 ("Digital Signature") for ACP certificates; this authentication method can be used with both RSA and ECDSA certificates, indicated by an ASN.1 object AlgorithmIdentifier.

The Digital Signature hash SHA2-512 MUST be supported (in addition to SHA2-256).

The IKEv2 Diffie-Hellman key exchange group 19 (256-bit random ECP), MUST be supported. Reason: ECC provides a similar security level to finite-field (MODP) key exchange with a shorter key length, so is generally preferred absent other considerations.

6.8.3.2. 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.

The requirements for ESP/IPsec/IKEv2 with GRE are the same as for native IPsec (see Section 6.8.3.1) except that IPsec transport mode and next protocol GRE (47) are to be negotiated. Tunnel mode is not required because of GRE. Traffic Selectors are:

TSi = (47, 0-65535, Initiator-IPv6-LL-addr ... Initiator-IPv6-LL-addr)

TSr = (47, 0-65535, Responder-IPv6-LL-addr ... Responder-IPv6-LL-addr)

If IKEv2 initiator and responder support IPsec over GRE, it will be preferred over native IPsec because of the way how IKEv2 negotiates transport mode (as used by this IPsec over GRE profile) versus tunnel mode as used by native IPsec (see [RFC7296], section 1.3.1). The ACP IPv6 traffic has to be carried across GRE according to [RFC7676].

6.8.4. ACP via DTLS

This document defines the use of ACP via DTLS, on the assumption that it is likely the first transport encryption supported in some classes of constrained devices: DTLS is commonly used in constrained devices when IPsec is not. Code-space on those devices may be also be too limited to support more than the minimum number of required protocols.

An ACP node announces its ability to support DTLS version 1.2 ([RFC6347]) compliant with the requirements defined in this document as an ACP secure channel protocol in GRASP through the "DTLS" objective-value in the "AN_ACP" objective (see Section 6.4).

To run ACP via UDP and DTLS, a locally assigned UDP port is used that is announced as a parameter in the GRASP AN_ACP objective to candidate neighbors. This port can also be any newer version of DTLS as long as that version can negotiate a DTLS v1.2 connection in the presence of an DTLS v1.2 only peer.

All ACP nodes supporting DTLS as a secure channel protocol MUST adhere to the DTLS implementation recommendations and security considerations of BCP 195, BCP 195 [RFC7525] except with respect to the DTLS version. ACP nodes supporting DTLS MUST support DTLS 1.2. They MUST NOT support older versions of DTLS.

Unlike for IPsec, no attempts are made to simplify the requirements of the BCP 195 recommendations because the expectation is that DTLS would be using software-only implementations where the ability to reuse of widely adopted implementations is more important than minimizing the complexity of a hardware accelerated implementation which is known to be important for IPsec.

DTLS v1.3 ([I-D.ietf-tls-dtls13]) is "backward compatible" with DTLS v1.2 (see section 1. of DTLS v1.3). A DTLS implementation supporting both DTLS v1.2 and DTLS v1.3 does comply with the above requirements of negotiating to DTLS v1.2 in the presence of a DTLS v1.2 only peer, but using DTLS v1.3 when both peers support it.

Version v1.2 is the MTI version of DTLS in this specification because

- * There is more experience with DTLS v1.2 across the spectrum of target ACP nodes.
- * Firmware of lower end, embedded ACP nodes may not support a newer version for a long time.
- * There are significant changes of DTLS v1.3, such as a different record layer requiring time to gain implementation and deployment experience especially on lower end, code space limited devices.
- * The existing BCP [RFC7525] for DTLS v1.2 may equally take longer time to be updated with experience from a newer DTLS version.
- * There are no significant use-case relevant benefits of DTLS v1.3 over DTLS v1.2 in the context of the ACP options for DTLS. For example, signaling performance improvements for session setup in DTLS v1.3 is not important for the ACP given the long-lived nature of ACP secure channel connections and the fact that DTLS connections are mostly link-local (short RTT).

Nevertheless, newer versions of DTLS, such as DTLS v1.3 have stricter security requirements and use of the latest standard protocol version is for IETF security standards in general recommended. Therefore, ACP implementations are advised to support all the newer versions of DTLS that can still negotiate down to DTLS v1.2.

[RFC-editor: if by the time of AUTH48, DTLS 1.3 would have evolved to be an RFC, then not only would the references to the DTLS v1.3 draft be changed to the RFC number, but that RFC is then going to be put into the normative list of references and the above paragraph is going to be amended to say: Implementations SHOULD support [DTLSv1.3-RFC]. This is not done right now, because there is no benefit in potentially waiting in RFC-editor queue for that RFC given how the text already lays out a non-normative desire to support DTLSv1.3.]

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.

6.8.5. ACP Secure Channel Profiles

As explained in the beginning of Section 6.6, 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.

An ACP node supporting the "baseline" profile MUST support IPsec natively and MAY support IPsec via GRE. An ACP node supporting the "constrained" profile 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 needs to 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.

IPsec natively with tunnel mode provides the shortest encapsulation overhead. GRE may be preferred by legacy implementations because the virtual interfaces required by ACP design in conjunction with secure channels have in the past more often been implemented for GRE than purely for native IPsec.

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.

6.9. GRASP in the ACP

6.9.1. 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 6.12).

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 6.9.2). 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.

6.9.2. 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 following explanation how ACP acts as the security and transport substrate for GRASP is visualized in Figure 9 below.

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. See Section 6.1 for TLS requirements. TLS mutual authentication MUST use the ACP domain membership check defined in (Section 6.2.3).

GRASP link-local multicast messages are targeted for a specific ACP virtual interface (as defined Section 6.13.5) 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.

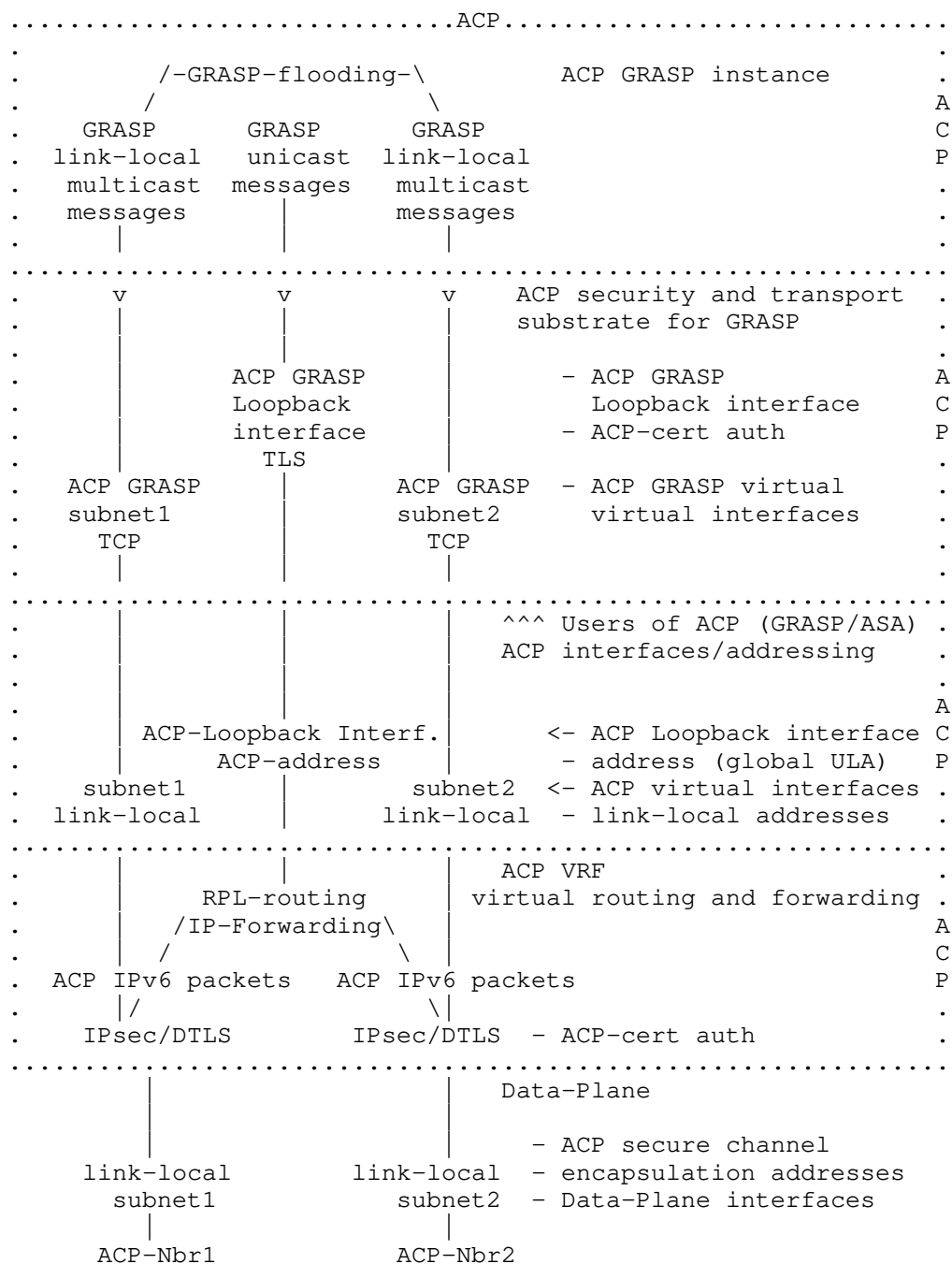


Figure 9: ACP as security and transport substrate for GRASP

6.9.2.1. 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 were to use just 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.

6.10. 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 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.

6.11. 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 creates a Loopback interface with an ACP network wide unique address (prefix) inside the ACP context (as explained in in Section 6.10). 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 10.1 for a discussion on merging domains.

Links inside the ACP only use link-local IPv6 addressing, such that each node's ACP only requires one routable address prefix.

6.11.1. 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 ULA 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 6.11.2 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 hosts and ACP addresses do therefore not represent end-users or groups of end-users. 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, NETCONF ...) are available in IPv6. See also [RFC8368] for how ACP could be made to interoperate with IPv4 only OAM.

Further explanation about the addressing and routing related reasons for the choice of the autonomous ACP addressing can be found in Section 6.13.5.1.

6.11.2. The ACP Addressing Base Scheme

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

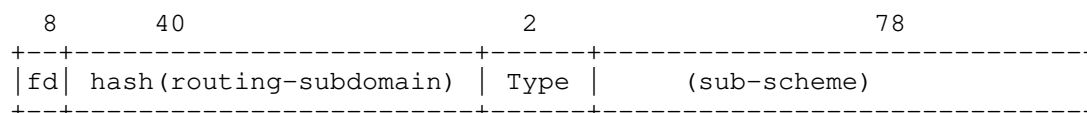


Figure 10: ACP Addressing Base Scheme

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 acp-node-name in the ACP certificates are the first 40-bits of the SHA256 hash of the routing subdomain from the same acp-node-name. In the example of Section 6.2.2, 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 other.
- * 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 acp-node-name 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 the resulting routing-subdomain hashes do not collide. For example, a large cooperation with its own private TA 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 6.11.7 and Appendix A.1 for further explanations why the following Sub-Addressing schemes are used and why multiple are necessary.

The following summarizes the addressing Sub-Schemes:

Type	Name	F-bit	Z	V-bits	Prefix
0x00	ACP-Zone	N/A	0	1 bit	/127
0x00	ACP-Manual	N/A	1	N/A	/64
0x01	ACP-VLong-8	0	N/A	8 bits	/120
0x01	ACP-VLong-16	1	N/A	16 bits	/112
0x10	Reserved / For future definition/allocation				
0x11	Reserved / For future definition/allocation				

Figure 11: Addressing Sub-Schemes

F-Bit and Z are two encoding fields explained below for the Sub-Schemes that introduce/use them. V-bits is the number of bits of addresses allocated to the ACP node. Prefix is the prefix the ACP node is announcing into the RPL routing protocol.

6.11.3. ACP Zone Addressing Sub-Scheme (ACP-Zone)

This sub-scheme is used when the Type field of the base scheme is 0x00 and the Z bit is 0x0.

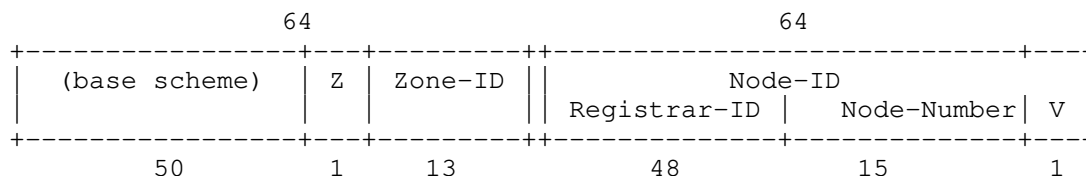


Figure 12: ACP Zone Addressing Sub-Scheme

The fields are defined as follows:

- * Type: MUST be 0x0.
- * Z: MUST be 0x0.
- * Zone-ID: A value for a network zone.
- * Node-ID: A unique value for each node.

The 64-bit Node-ID must be unique across the ACP domain for each node. It 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. One or more domain-wide unique identifiers of the ACP registrar can be used for this purpose. See Section 6.11.7.2.
- * Node-Number: Number to make the Node-ID unique. This can be sequentially assigned by the ACP Registrar owning the Registrar-ID.
- * 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 V field as all zero bits.

The ACP address set of the node includes addresses with any Zone-ID value and any V value. No two nodes in the same ACP can have the same Node-ID, but different Zone-IDs.

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.

It is RECOMMENDED that only Zone-ID 0 is used unless it is meant to be used in conjunction with operational practices for partial/incremental adoption of the ACP as described in Section 9.4.

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.

6.11.4. ACP Manual Addressing Sub-Scheme (ACP-Manual)

This sub-scheme is used when the Type field of the base scheme is 0x00 and the Z bit is 0x1.

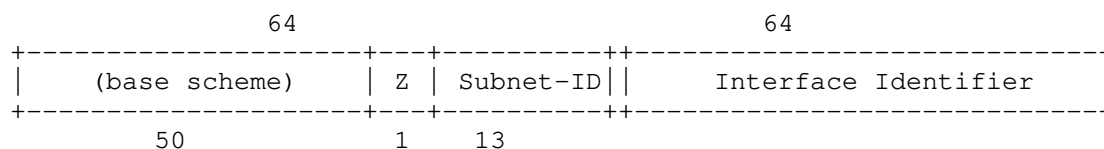


Figure 13: ACP Manual Addressing Sub-Scheme

The fields are defined as follows:

- * Type: MUST be 0x0.
- * Z: MUST be 0x1.
- * Subnet-ID: Configured subnet identifier.
- * 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 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 certificate MUST omit the acp-address field to indicate that their ACP 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. Therefore, the notion of V-bit many addresses assigned to the ACP nodes does not apply to this Sub-Scheme.

6.11.5. ACP Vlong Addressing Sub-Scheme (ACP-VLong-8/ACP-VLong-16)

This sub-scheme is used when the Type field of the base scheme is 0x01.

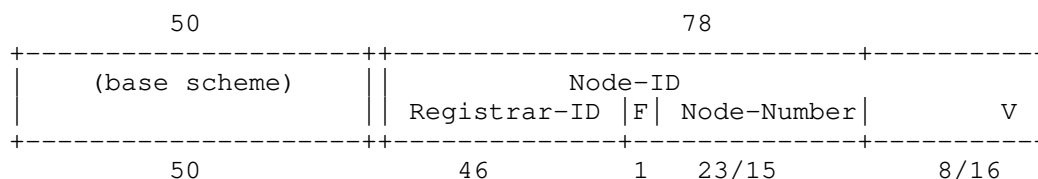


Figure 14: ACP Vlong Addressing Sub-Scheme

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:

- * F: format bit. This bit determines the format of the subsequent bits.
- * V: Virtualization bit: this is a field that is either 8 or 16 bits. For F=0, it is 8 bits, for F=1 it is 16 bits. The V bits are assigned by the ACP node. In the ACP certificate's ACP address Section 6.2.2, the V-bits are always set to 0.
- * Registrar-ID: To maximize Node-Number and V, the Registrar-ID is reduced to 46-bits. One or more domain-wide unique identifiers of the ACP registrar can be used for this purpose. See Section 6.11.7.2.
- * The Node-Number is unique to each ACP node. There are two formats for the Node-Number. When F=0, the node-number is 23 bits, for F=1 it is 15 bits. Each format of node-number is considered to be in a unique number space.

The F=0 bit format addresses 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.

The F=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 software 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.

6.11.6. 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.

6.11.7. ACP Registrars

ACP registrars are responsible to enroll candidate ACP nodes with ACP certificates and associated trust anchor(s). They are also responsible that an `acp-node-name` field is included in the ACP 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 certificates without dependencies against a (shared) TA (except during renewals of their own certificates).

ACP registrars are PKI registration authorities (RA) enhanced with the handling of the ACP certificate specific fields. They request certificates for ACP nodes from a Certification 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 can be given the necessary credentials to perform this RA function, such as credentials for the BRSKI connection to the CA for ANI registrars.

6.11.7.1. Use of BRSKI or other Mechanism/Protocols

Any protocols or mechanisms may be used by ACP registrars, as long as the resulting ACP certificate and TA certificate(s) allow to perform the ACP domain membership described in Section 6.2.3 with other ACP domain members, and meet the ACP addressing requirements for its `acp-node-name` 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 TA, using command line or web based commands on the candidate ACP node and TA to generate and sign the ACP certificate and configure certificate and TA 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-node-name` field because the rules do not depend on BRSKI but apply equally to any protocols/mechanisms an ACP registrar may use.

6.11.7.2. Unique Address/Prefix allocation

ACP registrars MUST NOT allocate ACP address prefixes to ACP nodes via the `acp-node-name` 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.

To support such unique address allocation, an ACP registrar MUST have one or more 46-bit identifiers unique across the ACP domain which is called the Registrar-ID. Allocation of Registrar-ID(s) to an ACP registrar can happen through OAM mechanisms in conjunction with some database / allocation orchestration.

ACP registrars running on physical devices with known globally unique EUI-48 MAC address(es) can use the lower 46 bits of those address(es) as unique Registrar-IDs without requiring any external signaling/configuration (the upper two bits, V and U are not uniquely assigned but functional). This approach is attractive for distributed, non-centrally administered, lightweight ACP registrar implementations. There is no mechanism to deduce from a MAC address itself whether it is actually uniquely assigned. Implementations need to consult additional offline information before making this assumption. For example by knowing that a particular physical product/MIC-chip is guaranteed to use globally unique assigned EUI-48 MAC address(es).

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 `acp-node-name` field (Section 6.2.2) with the appropriate information - ACP domain-name, ACP-address, and so on. If the ACP registrar uses BRSKI, it signals the ACP `acp-node-name` field to the Pledge via the `EST /csrattrs` command (see [I-D.ietf-anima-bootstrapping-keyinfra], section 5.9.2 - "EST CSR Attributes").

[RFC-Editor: please update reference to section 5.9.2 accordingly with latest BRSKI draft at time of publishing, or RFC]

6.11.7.3. 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 6.11.3), or a Vlong addressing sub-scheme prefix (see Section 6.11.5). The assigned ACP address prefix encoded in the `acp-node-name` field of the ACP 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 6.12) 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 certificate of the candidate ACP node, which typically contains a "serialNumber" attribute in the subject field distinguished name encoding that is often indicating the node's 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 Zone-ID 0.

ACP registrars that are aware of the IDevID certificate of a candidate ACP device SHOULD be able to choose the zone vs. Vlong sub-address scheme for ACP nodes based on the [X.520] "serialNumber" attribute in the subject field distinguished name encoding of the IDevID certificate, for example by the PID (Product Identifier) part which identifies the product type, or the complete "serialNumber". The PID for example could identify nodes that allow for specialized ASA requiring multiple addresses or non-autonomic VMs for services and those nodes could receive Vlong sub-address scheme ACP addresses.

In a simple allocation scheme, an ACP registrar remembers persistently across reboots its currently used Registrar-ID and for each addressing scheme (Zone with Zone-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.

If allocated addresses cannot be remembered by registrars, then it is necessary to either use a new value for the Register-ID field in the ACP addresses, or determine allocated ACP addresses from determining the addresses of reachable ACP nodes, which is not necessarily the set of all ACP nodes. Non-tracked ACP addresses can be reclaimed by revoking or not renewing their certificates and instead handing out new certificate with new addresses (for example with a new Registrar-ID value). Note that such strategies may require coordination amongst registrars.

6.11.7.4. Address/Prefix Persistence

When an ACP certificate is renewed or rekeyed via EST or other mechanisms, the ACP address/prefix in the `acp-node-name` 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 9.2.4 for an example.

ACP address information **SHOULD** also be maintained even after an ACP certificate did expire or failed. See Section 6.2.5.5 and Section 6.2.5.6.

6.11.7.5. Further Details

Section 9.2 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.

6.12. 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.6 for more details.

6.12.1. ACP 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 [I-D.ietf-roll-applicability-template].

6.12.1.1. Overview

RPL Packet Information (RPI) defined in [RFC6550], section 11.2 defines the data packet artefacts required or beneficial in forwarding of packets routed by RPL. This profile does not use RPI for better compatibility with accelerated hardware forwarding planes which most often does not support the Hop-by-Hop headers used for RPI, but also to avoid the overhead of the RPI header on the wire and cost of adding/removing them.

6.12.1.1.1. Single Instance

To avoid the need for RPI, the ACP RPL profile uses a simple destination prefix based routing/forwarding table. To achieve this, the profile 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.

This choice is a compromise. 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. See Appendix A.9.4 for more details.

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.

The benefit of this profile, especially compared to other IGPs 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.

6.12.1.1.2. Reconvergence

In RPL profiles where RPL Packet Information (RPI, see Section 6.12.1.13) is present, it is also used to trigger reconvergence when misrouted, for example looping, packets are recognized because of their RPI data. This helps to minimize RPL signaling traffic especially in networks without stable topology and slow links.

The ACP RPL profile instead relies on quick reconverging the DODAG by recognizing link state change (down/up) and triggering reconvergence signaling as described in Section 6.12.1.7. 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 6.12.1.7, loops caused by loss of RPL routing protocol signaling packets should be exceedingly rare.

In addition, there are a variety of mechanisms possible in RPL to further avoid temporary loops RECOMMENDED to be used for the ACPL RPL profile: DODAG Information Objects (DIOs) SHOULD be sent 2 or 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 through 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.

6.12.1.2. RPL Instances

Single RPL instance. Default RPLInstanceID = 0.

6.12.1.3. 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.

6.12.1.4. 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.

6.12.1.5. Path Metric

Use Hopcount according to [RFC6551]. Note that this is solely for diagnostic purposes as it is not used by the objective function.

6.12.1.6. Objective Function

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

rank_factor: Derived from link speed: $\leq 100\text{Mbps}$:
LOW_SPEED_FACTOR(5), else HIGH_SPEED_FACTOR(1)

This is a simple rank differentiation between typical "low speed" or "IoT" links that commonly max out at 100 Mbps and typical infrastructure links with speeds of 1 Gbps or higher. Given how the path selection for the ACP focusses only on reachability but not on path cost optimization, no attempts at finer grained path optimization are made.

6.12.1.7. DODAG Repair

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

Local Repair: As soon as link breakage is detected, the ACP node send No-Path DAO for all the targets that were reachable only via this link. As soon as link repair is detected, the ACP node validates if this link provides a better parent. If so, a new rank is computed by the ACP node and it sends new DIO that advertise the new rank. Then it sends a DAO with a new path sequence about itself.

When using ACP multi-access virtual interfaces, local repair can be triggered directly by peer breakage, see Section 6.13.5.2.2.

stretch_rank: none provided ("not stretched").

Data Path Validation: Not used.

Trickle: Not used.

6.12.1.8. Multicast

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

6.12.1.9. 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.

6.12.1.10. P2P communications

Not used.

6.12.1.11. IPv6 address configuration

Every ACP node (RPL node) announces an IPv6 prefix covering the addresses assigned to the ACP node via the AcpNodeName. The prefix length depends on the addressing sub-scheme of the acp-address, /127 for Zone Addressing Sub-Scheme and /112 or /120 for Vlong addressing sub-scheme. See Section 6.11 for more details.

Every ACP node MUST install a black hole (aka null) route if there are unused parts of the ACP address space assigned to the ACP node via its AcpnodeName. This is superseded by longer prefixes assigned to interfaces for the address space actually used by the node. For example, when the node has an ACP-VLong-8 address space, it installs a /120 black hole route. If it then for example only uses the ACP address (first address from the space), it would assign that address via a /128 address prefix to the ACP loopback interface (see Section 6.13.5.1). None of those longer prefixes are announced into RPL.

For ACP-Manual address prefixes configured on an ACP node, for example for ACP connect subnets (see Section 8.1.1), the node announces the /64 subnet prefix.

6.12.1.12. 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.

6.12.1.13. RPL Packet Information

RPI is not required in the ACP RPL profile for the following reasons.

One RPI option is the RPL Source Routing Header (SRH) [RFC6554] which is not necessary because the ACP RPL profile uses storing mode where each hop has the necessary next-hop forwarding information.

The simpler RPL Option header [RFC6553] is also not necessary in this profile, because it uses a single RPL instance and data path validation is also not used.

6.12.1.14. 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.

As this requirement places additional constraints on the Data-Plane functionality of the RPL root, it does not apply to "normal" nodes that are not configured to have special functionality (i.e., the administrative parameter from Section 6.12.1.12 has value 0b001). If the ACP network is degraded to the point where there are no nodes that could be configured as root, registrar, or ACP-connect nodes, it is possible that the RPL root (and thus the ACP as a whole) would be unable to detect traffic to unknown destinations. However, in the absence of nodes with administrative preference other than 0b001, there is also unlikely to be a way to get diagnostic information out of the ACP, so detection of traffic to unknown destinations would not be actionable anyway.

6.13. 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 6.12.

6.13.1. 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.

6.13.2. Addressing of Secure Channels

In order to be independent of the Data-Plane routing and addressing, the GRASP discovered 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.9.2.

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).

6.13.3. 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 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.

6.13.4. 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 6.6 allows the Decider (the node with the higher ACP address) to drop all but the desired ACP channels to the Follower - and the Follower will not re-try to build these secure channels from its side unless the Decider shows up with a previously unknown GRASP announcement (e.g., on a different link or with a different address announced in GRASP).

6.13.5. 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.

6.13.5.1. ACP loopback interfaces

For autonomous operations of the ACP, as described in Section 6 and Section 7, the ACP node uses the first address from the N bit ACP prefix ($N = 128 - \text{number of Vbits of the ACP address}$) assigned to the node. This address is assigned with an address prefix of N or larger to a loopback interface.

Other addresses from the prefix can be used by the ACP of the node as desired. The autonomous operations of the ACP does not require additional global scope IPv6 addresses, they are instead intended for ASA or non-autonomous functions. Non fully autonomic components of the ACP such as ACP connect interfaces (see Figure 16) may also introduce additional global scope IPv6 addresses on other types of interfaces into the ACP.

[RFC-Editor: please remove this paragraph: Note to reviewers: Please do not complain again about an obsolete RFC number in the following paragraph. The text should make it clear that the reference was chosen to indicate a particular point in time, but not to recommend/use a particularly obsolete protocol spec.]

The use of loopback interfaces for global scope addresses is common operational configuration practice on routers, for example in IBGP connections since BGP4 (see [RFC1654]) or earlier. The ACP adopts and automates this operational practice.

A loopback interface for use with the ACP as described above is an interface behaving according to [RFC6724] Section 4., paragraph 2: Packets sent by the host of the node from the loopback interface behave as if they are looped back by the interface so that they look as if they originated from the loopback interface, are then received by the node and forwarded by it towards the destination.

The word loopback only indicates this behavior, but not the actual name of the interface type chosen in an actual implementation. A loopback interface for use with the ACP can be a virtual/software construct without any associated hardware, or it can be a hardware interface operating in loopback mode.

A loopback interface used for the ACP MUST NOT have connectivity to other nodes.

The following reviews the reasons for the choice of loopback addresses for ACP addresses is based on the IPv6 address architecture and common challenges:

1. IPv6 addresses are assigned to interfaces, not nodes. IPv6 continues the IPv4 model that a subnet prefix is associated with one link, see [RFC4291], Section 2.1.
2. IPv6 implementations commonly do not allow assignment of the same IPv6 global scope address in the same VRF to more than one interface.
3. Global scope addresses assigned to interfaces that are connecting to other nodes (external interfaces) may not be stable addresses for communications because any such interface could fail due to reasons external to the node. This could render the addresses assigned to that interface unusable.
4. If failure of the subnet does not result in bringing down the interface and making the addresses unusable, it could result in unreachability of the address because the shortest path to the node might go through one of the other nodes on the same subnet which could equally consider the subnet to be operational even though it is not.
5. Many OAM service implementations on routers cannot deal with more than one peer address, often because they do already expect that a single loopback address can be used, especially to provide a stable address under failure of external interfaces or links.
6. Even when an application supports multiple addresses to a peer, it can only use one address for a connection at a time with the most widely deployed transport protocols TCP and UDP. While [RFC6824] solves this problem, it is not widely adopted for router OAM services implementations.
7. To completely autonomously assign global scope addresses to subnets connecting to other nodes, it would be necessary for every node to have an amount of prefix address space in the order of the maximum number of subnets that the node could connect to and then the node would have to negotiate with adjacent nodes across those subnets whose address space to use for each subnet.
8. Using global scope addresses for subnets between nodes is unnecessary if those subnets only connect routers, such as ACP secure channels, because they can communicate to remote nodes via their global scope loopback addresses. Using global scope addresses for those extern subnets is therefore wasteful for the address space and also unnecessarily increasing the size of routing and forwarding tables, which especially for the ACP is highly undesirable because it should attempt to minimize the per-node overhead of the ACP VRF.

9. For all these reasons, the ACP addressing schemes do not consider ACP addresses for subnets connecting ACP nodes.

Note that [RFC8402] introduces the term Node-SID to refer to IGP prefix segments that identify a specific router, for example on a loopback interface. An ACP loopback address prefix may similarly be called an ACP Node Identifier.

6.13.5.2. ACP virtual interfaces

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

Note that all the considerations described here are assuming point-to-point secure channel associations. Mapping multi-party secure channel associations such as [RFC6407] is out of scope.

6.13.5.2.1. ACP point-to-point virtual interfaces

In this option, 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.

When the secure channel protocol determines a peer to be dead, this SHOULD result in indicating link breakage to trigger RPL DODAG repair, see Section 6.12.1.7.

6.13.5.2.2. ACP multi-access virtual interfaces

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.

When the secure channel protocol determines a peer to be dead for a secure channel mapped into an ACP multi-access virtual interface, this SHOULD result in signaling breakage of that peer to RPL, so it can trigger RPL DODAG repair, see Section 6.12.1.7.

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 per-peer, point-to-point virtual interfaces, one to Bob and one to Carol, Alice's ACP GRASP will send two copies of multicast GRASP messages: One to Bob and one to Carol. If Alice's ACP emulates a LAN via a multipoint virtual interface, Alice's ACP GRASP will send one packet to that interface and the ACP multipoint virtual interface will replicate the packet to each secure channel, one to Bob, one to Carol. 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 6.9.2 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.9.4.

Care needs to be taken when creating multi-access ACP virtual interfaces across ACP secure channels between ACP nodes in different domains or routing subdomains. If for example future inter-domain ACP policies are defined as "peer-to-peer" policies, it is easier to create ACP point-to-point virtual interfaces for these inter-domain secure channels.

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

7.1. Why (Benefits of ACP on L2 switches)

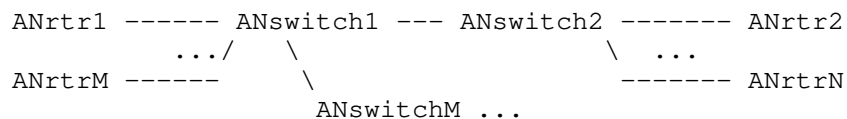


Figure 15: Topology with L2 ACP switches

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 (instead of being flooded across all ports of the VLAN to which the port belongs).

When Ports/Interfaces across which the ACP is expected to operate in an ACP-aware L2-switch or L2/L3-switch/router are L2-bridged, packets for the ALL_GRASP_NEIGHBORS multicast address MUST never be forward between these ports. If MLD snooping is used, it MUST be prohibited from bridging packets for the ALL_GRASP_NEIGHBORS IPv6 multicast address.

On hybrid L2/L3 switches, multiple L2 ports are assigned to a single L3 VLAN interface. With the aforementioned changes for DULL GRASP, ACP can simply operate on the L3 VLAN interfaces, so no further (hardware) forwarding changes are required to make ACP operate on L2 ports. This is possible because the ACP secure channel protocols only use link-local IPv6 unicast packets, and these packets will be sent to the correct L2 port towards the peer by the VLAN logic of the device.

This is sufficient when p2p ACP virtual interfaces are established to every ACP peer. When it is desired to create multi-access ACP virtual interfaces (see Section 6.13.5.2.2), it is REQUIRED not to coalesce all the ACP secure channels on the same L3 VLAN interface, but only all those on the same L2 port.

If VLAN tagging is used, then all the above described logic only applies to untagged GRASP packets. For the purpose of ACP neighbor discovery via GRASP, no VLAN tagged packets SHOULD be sent or received. In a hybrid L2/L3 switch, each VLAN would therefore only create ACP adjacencies across those ports where the VLAN is carried untagged.

In result, the simple logic is that ACP secure channels would operate over the same L3 interfaces that present a single flat bridged network across all routers, but because DULL GRASP is separated on a per-port basis, no full mesh of ACP secure channels is created, but only per-port ACP secure channels to per-port L2-adjacent ACP node neighbors.

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.

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

8.1.1.1. 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 needs to 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 SHOULD 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 ACP secure channel that is encapsulating the traffic.

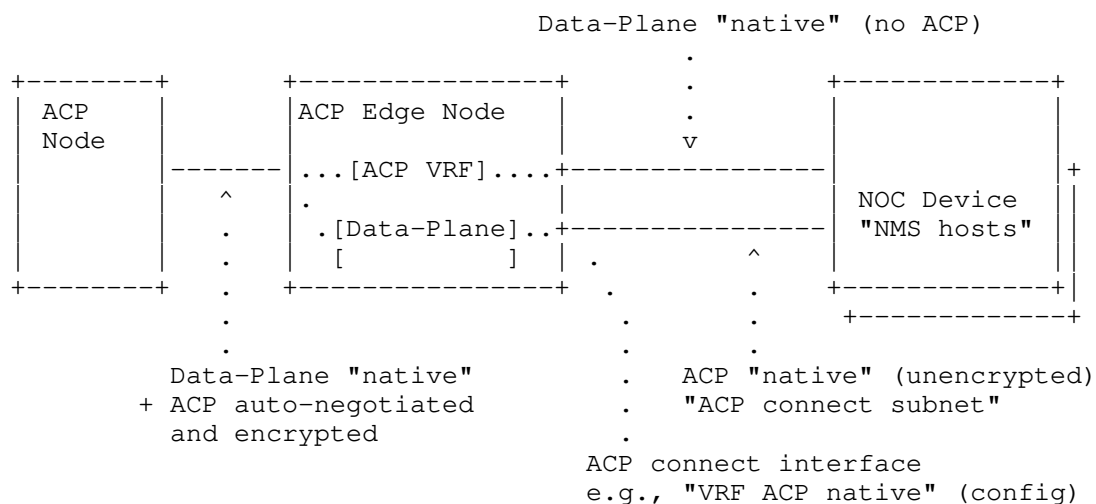


Figure 16: ACP connect

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 needs to 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.

Physical controlled/secured means that attackers can gain no access to the physical device hosting the ACP Edge Node, the physical interfaces and links providing the ACP connect link nor the physical devices hosting the NOC Device. In a simple case, ACP Edge node and NOC Device are co-located in an access controlled room, such as a NOC, to which attackers cannot gain physical access.

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 6.11.4), 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, the ACP Edge node MUST only forward the packet into the ACP if the packet has an IPv6 source address from that interface (this is sometimes called "RPF filtering"). This filtering rule MAY be changed through administrative measures. The more any such administrative action enable reachability of non ACP nodes to the ACP, the more this may cause security issues.

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.9.5.

ACP Edge nodes SHOULD have a configurable option to prohibit packets with RPI headers (see Section 6.12.1.13 across an ACP connect interface. These headers are outside the scope of the RPL profile in this specification but may be used in future extensions of this specification.

8.1.2. Software Components

The previous section assumed that ACP Edge node and NOC devices are separate physical devices and the ACP connect interface is a physical network connection. This section discusses the implication when these components are instead software components running on a single physical device.

The ACP connect mechanism cannot 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 be 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 towards the ACP on the ACP connect subnet, providing the (auto-)configured prefix for the ACP connect subnet and (auto-)configured routes into the ACP to NMS hosts and/or software components.

The ACP edge node uses the Route Information Option (RIO) of RFC4191 to announce aggregated prefixes for address prefixes used in the ACP (with normal RIO lifetimes. In addition, the ACP edge node also uses a RIO to announce the default route (:::/0) with a lifetime of 0.

These RIOs allow to connect Type C hosts to the ACP via an ACP connect subnet on one interface and another network (Data Plane / NMS network) on the same or another interface of the Type C host, relying on other routers than the ACP edge node. The RIOs ensure that these hosts will only route the prefixes used in the ACP to the ACP edge node.

Type A/B host ignore the RIOs and will consider the ACP node to be their default router for all destination. This is sufficient when type A/B hosts only need to connect to the ACP but not to other networks. Attaching Type A/B hosts to both the ACP and other networks, requires either explicit ACP prefix route configuration on the Type A/B hosts or the combined ACP/Data-Plane interface on the ACP edge node, see Section 8.1.4.

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)

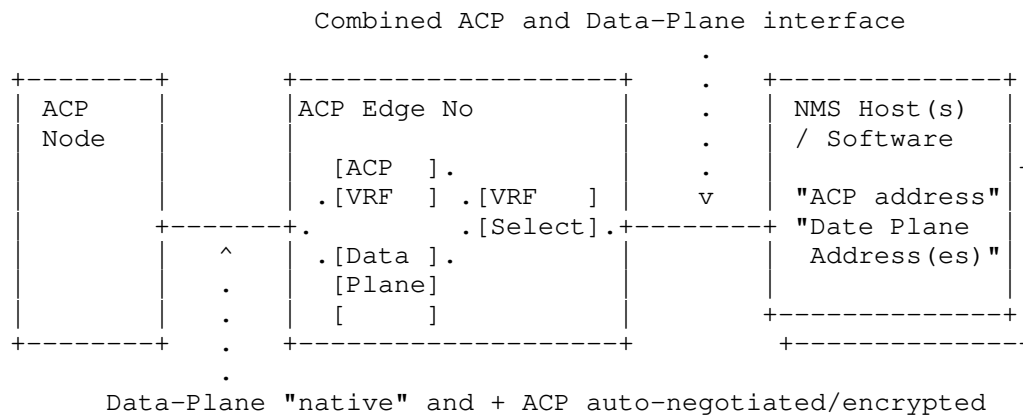


Figure 17: VRF select

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, or it may be insufficient to support [RFC4191] Type A or B host (see Section 8.1.3).

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 requirements for devices connected via ACP connect is that those are equivalently (if not better) secured against attacks than ACP nodes that do not use ACP connect and run only software that is equally (if not better) protected, known (or trusted) not to be malicious and accordingly designed to isolate access to the ACP against external equipment.

The difference in security is that cryptographic security of the ACP secure channel is replaced by required physical security/control of the network connection between an ACP edge node and the NMS or other host reachable via the ACP connect interface. See Section 8.1.1.

When using "Combined ACP and Data-Plane Interfaces", care has to 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 SHOULD 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. Connecting ACP islands over Non-ACP L3 networks (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 18: 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.

Tunneling using an insecure tunnel encapsulation increases on average the risk of a MITM downgrade attack somewhere along the underlay path that blocks all but the most easily attacked ACP secure channel option. ACP nodes supporting tunneled remote ACP Neighbors SHOULD support configuration on such tunnel interfaces to restrict or explicitly select the available ACP secure channel protocols (if the ACP node supports more than one ACP secure channel protocol in the first place).

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. 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 9.1 describes recommended operator diagnostics capabilities of ACP nodes.
- * Section 9.2 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 9.3 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.

9.1. 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.

Basic standardized diagnostics would require support for (yang) models representing the complete (auto-)configuration and operational state of all components: GRASP, ACP and the infrastructure used by them: TLS/DTLS, IPsec, certificates, TA, 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 simplest 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 9.3 for a discussion of such commands).
- * 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)?.

- * Does the node have keying material - domain certificate, TA certificates, ...>
- * If no keying material and ANI is supported/enabled, check the state of BRSKI (not detailed in this example).
- * Check the validity of the domain certificate:
 - Does the certificate validate against the TA?
 - 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-node-name field?
- * Was the ACP VRF successfully created?
- * 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?

- * Is the interface physically up? Does it have an IPv6 link-local address?
- * Is it enabled for ACP?
- * Do we successfully send DULL GRASP messages to the interface (link layer errors)?
- * 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.
- * Do we see the ACP objective in any DULL GRASP message from that interface? Diagnose the supported secure channel methods.
- * Do we know the MAC address of the neighbor with the ACP objective? If not, diagnose SLAAC/ND state.
- * When did we last attempt to build an ACP secure channel to the neighbor?
- * 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:
 - o 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).

- o The ACP certificate membership check (Section 6.2.3) fails:
 - + The neighbor's certificate is not signed directly or indirectly by one of the nodes TA. Provide diagnostics which TA it has (can identify whom the device belongs to).
 - + The neighbor's certificate does not have the same domain (or no domain at all). Diagnose domain-name and potentially other cert info.
 - + The neighbor's certificate has been revoked or could not be authenticated by OCSP.
 - + The neighbor's 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?

- * 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 IDevID certificate. 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. 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 be considered to be too insecure to make this information available unprotected to all possible neighbors.

An "interested adjacent party" can always determine the IDevID certificate of a BRSKI pledge by behaving like a BRSKI proxy/registrar. Therefore, the IDevID certificate 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".

9.1.1. Secure Channel Peer diagnostics

When using mutual certificate authentication, the TA certificate is not required to be signaled explicitly because its hash is sufficient for certificate chain validation. In the case of ACP secure channel setup this leads to limited diagnostics when authentication fails because of TA mismatch. For this reason, Section 6.8.2 recommends to also include the TA certificate in the secure channel signaling. This should be possible to do without protocol modifications in the security association protocols used by the ACP. For example, while [RFC7296] does not mention this, it also does not prohibit it.

One common deployment use case where the diagnostic through the signaled TA of a candidate peer is very helpful are multi-tenant environments such as office buildings, where different tenants run their own networks and ACPs. Each tenant is given supposedly disjoint L2 connectivity through the building infrastructure. In these environments there are various common errors through which a device may receive L2 connectivity into the wrong tenants network.

While the ACP itself is not impacted by this, the Data-Plane to be built later may be impacted. Therefore, it is important to be able to diagnose such undesirable connectivity from the ACP so that any autonomic or non-autonomic mechanisms to configure the Data-Plane can accordingly treat such interfaces. The information in the TA of the peer can then ease troubleshooting of such issues.

Another example case is the intended or accidental re-activation of equipment whose TA certificate has long expired, such as redundant gear taken from storage after years.

A third example case is when in a mergers & acquisition case ACP nodes have not been correctly provisioned with the mutual TA of previously disjoint ACP. This is assuming that the ACP domain names were already aligned so that the ACP domain membership check is only failing on the TA.

A fourth example case is when multiple registrars were set up for the same ACP but without correctly setting up the same TA. For example, when registrars support to also be CA themselves but are misconfigured to become TA instead of intermediate CA.

9.2. ACP Registrars

As described in Section 6.11.7, 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 certificate into an ACP domain via any appropriate mechanism/protocol, automated or not.

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

9.2.1. 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 TA is required. Typically, this is a root CA. 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 TA that has no ACP connectivity, 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 certificate. NETCONF ZeroTouch ([RFC8572]) 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 TA and other sources of CRL information.

9.2.2. Registrar Parameter

The interactions of an ACP registrar outlined Section 6.11.7 and Section 9.2.1 above depend on the following parameters:

- * A URL to the TA and credentials so that the ACP registrar can let the TA sign candidate ACP node 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 /120 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 IDevID certificate as in BRSKI. The ACP registrar may have a whitelist or blacklist of devices [X.520] "serialNumbers" attribute in the subject field distinguished name encoding from their IDevID certificate.
- * 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 certificate of candidate ACP nodes (as defined in BRSKI).

9.2.3. 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 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-node-name from the ACP nodes current ACP certificate and maintain this information so that the ACP node maintains its ACP address prefix. In EST renewal/rekeying, the ACP nodes current ACP 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 TA.
2. Recovery from a compromised ACP registrar is difficult. When an ACP registrar is compromised, it can insert for example a conflicting acp-node-name 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 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.

9.2.4. 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 TA whenever an ACP node is enrolled, and reduces the need for connectivity of such an ACP registrar to a TA 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 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 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 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.

9.2.5. 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 policy per "serialNumber" attribute in the subject field distinguished name encoding 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.

9.3. 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.

9.3.1. 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.

9.3.2. 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 9.3.2.2).

One of the common problems of remote management is for the operator or SDN controller to cut its own connectivity to the remote node by a configuration impacting its own management connection into the node. The ACP itself should have no dedicated configuration other than aforementioned enablement of the ACP on brownfield ACP nodes. This leaves configuration that cannot distinguish between ACP and Data-Plane as sources of configuration mistakes as these commands will impact the ACP even though they should only impact the Data-Plane.

The one ubiquitous type of commands that do this on many type of routers are interface "down" commands/configurations. When such a command is applied to the interface through which the ACP provides access for remote management it would cut the remote management connection through the ACP because, as outlined above, the "down" commands typically impact the physical layer too and not only the Data-Plane services.

To provide ACP/ANI resilience against such operator misconfiguration, 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 9.3.1.

An example of non-ACP but ANI traffic that should be permitted to pass even in "admin-down" state is BRSKI enrollment traffic between BRSKI pledge and a BRSKI proxy.

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.

9.3.2.1. 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.

9.3.2.2. 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 9.1 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.

9.3.2.3. 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

diagnostic that automatically terminates in a predetermined manner without dependence on external input ensuring the link will become operational again.

9.3.2.4. 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.

9.3.3. 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".

9.3.4. Which interfaces to auto-enable?

(Section 6.4) 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.

9.3.5. 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 an 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".

9.3.5.1. 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 simplest 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 BRSKI and MASA operations could also be an unlikely but then critical security issue. If an attacker could impersonate the operator and register as the operator at the MASA or otherwise get hold of vouchers and can get enough physical access to the network so pledges would register to an attacking registrar, then the attacker could gain access to the ACP, and through the ACP gain access to the Data-Plane.

In networks where the operator explicitly wants to enable the ANI this could not happen, because the operator would create a BRSKI registrar that would discover attack attempts, and the operator would be setting up his registrar with the MASA. 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.

9.3.5.2. Greenfield nodes

An ACP "greenfield" node is one that does not have any prior configuration and that can be bootstrapped into the ACP across the network. To support greenfield nodes, ACP as described in this document needs to be combined with a bootstrap protocol/mechanism that will enroll the node with the ACP keying material - ACP certificate and TA. For ANI nodes, this protocol/mechanism is BRSKI.

When such a node is powered on and determines it is in greenfield condition, it enables the bootstrap protocol(s)/mechanism(s), and once the ACP keying material is enrolled, greenfield state ends and the ACP is started. When BRSKI is used, the node's state reflects this by setting "ANI enable" upon determination of greenfield state at power on.

ACP greenfield nodes that in the absence of ACP would have their interfaces in "down" state SHOULD set all native interfaces into "admin down" state and only permit Data-Plane traffic required for the bootstrap protocol/mechanisms.

ACP greenfield state ends either through successful enrolment of ACP keying material (certificate, TA) or detection of a permitted termination of ACP greenfield operations.

ACP nodes supporting greenfield operations MAY want to provide backward compatibility with other forms of configuration/provisioning, especially when only a subset of nodes are expected to be deployed with ACP. Such an ACP node SHOULD observe attempts to

provision/configure the node via interfaces/methods that traditionally indicate physical possession of the node, such as a serial or USB console port or a USB memory stick with a bootstrap configuration. When such an operation is observed before enrollment of the ACP keying material has completed, the node SHOULD put itself into the state the node would have been in, if ACP/ANI was disabled at boot (terminate ACP greenfield operations).

When an ACP greenfield node enables multiple automated ACP or non-ACP enrollment/bootstrap protocols/mechanisms in parallel, care must be taken not to terminate any protocol/mechanism before another one has succeeded to enroll ACP keying material or has progressed to a point where it is permitted to be a termination reason for ACP greenfield operations.

Highly secure ACP greenfield nodes may not permit any reason to terminate ACP greenfield operations, including physical access.

Nodes that claim to support ANI greenfield operations SHOULD NOT enable in parallel to BRSKI any enrollment/bootstrap protocol/mechanism that allows Trust On First Use (TOFU, [RFC7435]) over interfaces other than those traditionally indicating physical possession of the node. Protocols/mechanisms with published default username/password authentication are considered to suffer from TOFU. Securing the bootstrap protocol/mechanism by requiring a voucher ([RFC8366]) can be used to avoid TOFU.

In summary, the goal of ACP greenfield support is to allow remote automated enrollment of ACP keying materials, and therefore automated bootstrap into the ACP and to prohibit TOFU during bootstrap with the likely exception (for backward compatibility) of bootstrapping via interfaces traditionally indicating physical possession of the node.

9.3.6. 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 (future) property in the certificate (e.g., in the acp-node-name extension field): 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.

9.3.7. 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.

9.4. Partial or Incremental adoption

The ACP Zone Addressing Sub-Scheme (see Section 6.11.3) allows incremental adoption of the ACP in a network where ACP can be deployed on edge areas, but not across the core that is connecting those edges.

In such a setup, each edge network, such as a branch or campus of an enterprise network has a disjointed ACP to which one or more unique Zone-IDs are assigned: ACP nodes registered for a specific ACP zone have to receive ACP Zone Addressing Sub-scheme addresses, for example by virtue of configuring for each such zone one or more ACP Registrars with that Zone-ID. All the Registrars for these ACP Zones need to get ACP certificates from CAs relying on a common set of TA and of course the same ACP domain name.

These ACP zones can first be brought up as separate networks without any connection between them and/or they can be connected across a non-ACP enabled core network through various non-autonomic operational practices. For example, each separate ACP Zone can have an edge node that is a layer 3 VPN PE (MPLS or IPv6 layer 3 VPN), where a complete non-autonomic ACP-Core VPN is created by using the ACP VRFs and exchanging the routes from those ACP VRFs across the VPNs non-autonomic routing protocol(s).

While such a setup is possible with any ACP addressing sub-scheme, the ACP-Zone Addressing sub-scheme makes it easy to configure and scalable for any VPN routing protocols because every ACP zone would only need to indicate one or more /64 ACP Zone Addressing prefix routes into the ACP-Core VPN as opposed to routes for every individual ACP node as required in the other ACP addressing schemes.

Note that the non-autonomous ACP-Core VPN would require additional extensions to propagate GRASP messages when GRASP discovery is desired across the zones.

For example, one could set up on each Zone edge router a remote ACP tunnel to a GRASP hub. The GRASP hub could be implemented at the application level and could run in the NOC of the network. It would serve to propagate GRASP announcements between ACP Zones and/or generate GRASP announcements for NOC services.

Such a partial deployment may prove to be sufficient or could evolve to become more autonomous through future standardized or non-standardized enhancements, for example by allowing GRASP messages to be propagated across the layer 3 VPN, leveraging for example L3VPN Multicast support.

Finally, these partial deployments can be merged into a single contiguous complete autonomous ACP (given appropriate ACP support across the core) without changes in the crypto material, because the node's ACP certificates are from a single ACP.

9.5. 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 relies 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 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 6.2.5), but their configuration is not.

- * When non-ACP capable nodes such as pre-existing NMS need to be physically connected to the ACP, the ACP node to which they attach needs to be configured with ACP-connect according to Section 8.1. It is also possible to use that single physical connection to connect both to ACP and the Data-Plane of the network as explained in Section 8.1.4.
- * When devices are not autonomically bootstrapped, explicit configuration to enable the ACP needs to be applied. See Section 9.3.
- * When the ACP needs to be extended across interfaces other than L2, the ACP as defined in this document cannot autodiscover candidate neighbors automatically. Remote 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.

10. Summary: Benefits (Informative)

10.1. 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 peer's (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 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 Certification 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 9.2.4. 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 Registrar's 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 TA, a re-merge will be smooth.

Merging two networks with different TA requires the ACP nodes to trust the union of TA. As long as the routing-subdomain hashes are different, the addressing will not overlap. Accidentally, overlaps will only happen in the unlikely event of a 40-bit hash collision in SHA256 (see Section 6.11). 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).

10.2. Self-Protection Properties

10.2.1. 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.

Attacker will not be able to join the ACP unless they have a valid ACP certificate. On-path attackers without a valid ACP certificate cannot inject packets into the ACP due to ACP secure channels. They can also not decrypt ACP traffic except if they can crack the encryption. They can attempt behavioral traffic analysis on the encrypted ACP traffic.

The degree to which compromised ACP nodes can impact the ACP depends on the implementation of the ACP nodes and their impairment. When an attacker has only gained administrative privileges to configure ACP nodes remotely, the attacker can disrupt the ACP only through one of the few configuration options to disable it, see Section 9.3, or by configuring of non-autonomic ACP options if those are supported on the impaired ACP nodes, see Section 8. Injecting or extracting traffic into/from an impaired ACP node is only possible when an impaired ACP node supports ACP connect (see Section 8.1) and the attacker can control traffic into/from one of the ACP nodes interfaces, such as by having physical access to the ACP node.

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 ([RFC3596]), DHCPv6 ([RFC3315]), syslog ([RFC3164]), RADIUS ([RFC2865]), Diameter ([RFC6733]), TACACS ([RFC1492]), IPFIX ([RFC7011]), Netflow ([RFC3954]) – just to name a few. Not all of these protocol references are necessarily the latest version of protocols but versions that are still widely deployed.

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 send IPv6 link-local packets. The only exceptions 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.

10.2.2. From the inside

The security model of the ACP is based on trusting all members of the group of nodes that receive an ACP 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 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 ACP 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.9.8 for further considerations how to avoid and deal with compromised nodes.

10.3. 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 only limited 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 (except for Section 8 and Section 9.2) 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 aspects 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.

11. Security Considerations

A set of ACP nodes with ACP certificates for the same ACP domain and with ACP functionality enabled is automatically "self-building": The ACP is automatically established between neighboring ACP nodes. It is also "self-protecting": The ACP secure channels are authenticated and encrypted. No configuration is required for this.

The self-protecting property does not include workarounds for non-autonomic components as explained in Section 8. See Section 10.2 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.
- * ACP nodes receive their certificates from ACP registrars. These ACP registrars are security critical dependencies of the ACP: Procedures and protocols for ACP registrars are outside the scope of this specification as explained in Section 6.11.7.1, only requirements against the resulting ACP certificates are specified.
- * Every ACP registrar (for enrollment of ACP certificates) and ACP EST server (for renewal of ACP certificates) is a security critical entity and its protocols are security critical protocols. Both need to be hardened against attacks, similar to a CA and its protocols. A malicious registrar can enroll malicious nodes to an ACP network (if the CA delegates this policy to the registrar) or break ACP routing for example by assigning duplicate ACP address assignment to ACP nodes via their ACP certificates.
- * ACP nodes that are ANI nodes rely on BRSKI as the protocol for ACP registrars. For ANI type ACP nodes, the security considerations of BRSKI apply. It enables automated, secure enrollment of ACP certificates.
- * BRSKI and potentially other ACP registrar protocol options require that nodes have an (X.509v3 based) IDevID. IDevIDs are an option for ACP registrars to securely identify candidate ACP nodes that should be enrolled into an ACP domain.

- * For IDevIDs to securely identify the node to which it IDevID is assigned, the node needs to (1) utilize hardware support such as a Trusted Platform Module (TPM) to protect against extraction/cloning of the private key of the IDevID and (2) a hardware/software infrastructure to prohibit execution of non-authenticated software to protect against malicious use of the IDevID.
- * Like the IDevID, the ACP certificate should equally be protected from extraction or other abuse by the same ACP node infrastructure. This infrastructure for IDevID and ACP certificate is beneficial independent of the ACP registrar protocol used (BRSKI or other).
- * Renewal of ACP certificates requires support for EST, therefore the security considerations of [RFC7030] related to certificate renewal/rekeying and TP renewal apply to the ACP. EST security considerations when using other than mutual certificate authentication do not apply nor do considerations for initial enrollment via EST apply, except for ANI type ACP nodes because BRSKI leverages EST.
- * A malicious ACP node could declare itself to be an EST server via GRASP across the ACP if malicious software could be executed on it. CA should therefore authenticate only known trustworthy EST servers, such as nodes with hardware protections against malicious software. When Registrars use their ACP certificate to authenticate towards a CA, the id-kp-cmcRA [RFC6402] extended key usage attribute allows the CA to determine that the ACP node was permitted during enrollment to act as an ACP registrar. Without the ability to talk to the CA, a malicious EST server can still attract ACP nodes attempting to renew their keying material, but they will fail to perform successful renewal of a valid ACP certificate. The ACP node attempting to use the malicious EST server can then continue to use a different EST server, and log a failure against a malicious EST server.
- * Malicious on-path ACP nodes may filter valid EST server announcements across the ACP, but such malicious ACP nodes could equally filter any ACP traffic such as the EST traffic itself. Either attack requires the ability to execute malicious software on an impaired ACP node though.
- * In the absence of malicious software injection, an attacker that can misconfigure an ACP node which is supporting EST server functionality could attempt to configure a malicious CA. This would not result in the ability to successfully renew ACP certificates, but it could result in DoS attacks by becoming an EST server and making ACP nodes attempting their ACP certificate renewal via this impaired ACP node. This problem can be avoided when the EST server implementation can verify that the CA configured is indeed providing renewal for certificates of the node's ACP. The ability to do so depends on the EST-Server to CA protocol, which is outside the scope of this document.

In summary, attacks against the PKI/certificate dependencies of the ACP can be minimized by a variety of hardware/software components including options such as TPM for IDevID/ACP-certificate, prohibitions against execution of non-trusted software and design aspects of the EST Server functionality for the ACP to eliminate configuration level impairment.

Because ACP peers select one out of potentially more than one mutually supported ACP secure channel protocols via the approach described in Section 6.6, ACP secure channel setup is subject to downgrade attacks by MITM attackers. This can be discovered after such an attack by additional mechanisms described in Appendix A.9.9. Alternatively, more advanced channel selection mechanisms can be devised. [RFC-Editor: Please remove the following sentence]. See [ACPDRAFT] Appendix B.1. Both options are out of scope of this document.

The security model of the ACP as defined in this document is tailored for use with private PKI. The TA of a private PKI provide the security against maliciously created ACP certificates to give access to an ACP. Such attacks can create fake ACP certificates with correct looking AcpNodeNames, but those certificates would not pass the certificate path validation of the ACP domain membership check (see Section 6.2.3, point 2).

[RFC-Editor: please remove the following paragraph].

Using public CA is out of scope of this document. See [ACPDRAFT], Appendix B.3 for further considerations.

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. New protocol/services run across the ACP should therefore use end-to-end authentication inside the ACP. This is already done by GRASP as specified in this document.

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 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.9.8).

Higher layer service built using ACP 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 (CLI, NETCONF). This can be done through ACP certificates by differentiating them and introduce roles. See Appendix A.9.5.

Operators and provisioning software developers need to be aware of how the provisioning/configuration of network devices impacts the ability of the operator / provisioning software to remotely access the network nodes. By using the ACP, most of the issues of configuration/provisioning caused loss of connectivity for remote provisioning/configuration will be eliminated, see Section 6. Only few exceptions such as explicit physical interface down configuration will be left Section 9.3.2.

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 6.2.2. This allows even verification of ownership of a peer's 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 6.9.2.

ACP provides for secure, resilient zero-touch discovery of EST servers for certificate renewal. See Section 6.2.5.

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 6.6.

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 6.13.2.

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 6.8.5.

Section 7.2 describes how to implement a routed ACP topology operating on what effectively is a large bridge-domain when using L3/L2 routers that operate at L2 in the Data-Plane. In this case, the ACP is subject to much higher likelihood of attacks by other nodes "stealing" L2 addresses than in the actual routed case. Especially when the bridged network includes non-trusted devices such as hosts. This is a generic issue in L2 LANs. L2/L3 devices 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 MAC/IPv6 source addresses from wrong ports. This type of function needs to be enabled to prohibit DoS attacks and specifically to protect the ACP. 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.

12. IANA Considerations

This document defines the "Autonomic Control Plane".

For the ANIMA-ACP-2020 ASN.1 module, IANA is asked to register value IANA1 for "id-mod-anima-acpnode-name-2020" in the "SMI Security for PKIX Module Identifier" (1.3.6.1.5.5.7.0) registry.

For the otherName / AcpNodeName, IANA is asked to register a value for IANA2 for id-on-AcpNodeName in the "SMI Security for PKIX Other Name Forms" (1.3.6.1.5.5.7.8) registry.

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 6.4.

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 6.2.5.

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 be considered to be automatically registered in that registry for the same service for which a <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 Section 6.11.3)

1: ACP Vlong Addressing Sub-Scheme (ACP RFC Section 6.11.5) / ACP Manual Addressing Sub-Scheme (ACP RFC Section 6.11.4)

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.

Many thanks Ben Kaduk, Roman Danyliv and Eric Rescorla for their thorough SEC AD reviews, Russ Housley and Erik Kline for their reviews and to Valery Smyslov, Tero Kivinen, Paul Wouters and Yoav Nir for review of IPsec and IKEv2 parameters and helping to understand those and other security protocol details better. Thanks for Carsten Borman for CBOR/CDDL help.

Further input, review or suggestions were received from: Rene Struik, Benoit Claise, William Atwood and Yongkang Zhang.

14. Contributors

For all things GRASP including validation code, ongoing document text support and technical input.

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

Email: brian.e.carpenter@gmail.com

For RPL contributions and all things BRSKI/bootstrap including validation code, ongoing document text support and technical input.

Michael C. Richardson
Sandelman Software Works

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

For the RPL technology choices and text.

Pascal Thubert
Cisco Systems, Inc
Building D
45 Allee des Ormes - BP1200
06254 MOUGINS - Sophia Antipolis
France

Phone: +33 497 23 26 34
Email: pthubert@cisco.com

15. Change log [RFC-Editor: Please remove]

This document was developed on <https://github.com/anima-wg/autonomic-control-plane/tree/master/draft-ietf-anima-autonomic-control-plane>. That github repository also contains the document review/reply emails.

15.1. Summary of changes since entering IESG review

This text replaces the prior changelog with a summary to provide guidance for further IESG review.

Please see revision -21 for the individual changelogs of prior versions .

15.1.1. Reviews (while in IESG review status) / status

This document entered IESG review with version -13. It has since seen the following reviews:

IESG: Original owner/Yes: Terry Manderson (INT).

IESG: No Objection: Deborah Brungard (RTG), Alissa Cooper (GEN), Warren Kumari (OPS), Mirja Kuehlewind (TSV), Alexey Melnikov (ART), Adam Roach (ART).

IESG: No Objection, not counted anymore as they have left IESG: Ben Campbell (ART), Spencer Dawkins (TSV).

IESG: Open DISCUSS hopefully resolved by this version: Eric Rescorla (SEC, left IESG), Benjamin Kaduk (SEC).

Other: Michael Richardson (WG), Brian Carpenter (WG), Pascal Thubert (WG), Frank Xialiang (WG), Elwyn Davies (GEN), Joel Halpern (RTGdir), Yongkang Zhang (WG), William Atwood (WG).

15.1.2. BRSKI / ACP registrar related enhancements

Only after ACP entered IESG review did it become clear that the in-progress BRSKI document would not provide all the explanations needed for ACP registrars as expected earlier by ACP authors. Instead, BRSKI will only specify a subset of required ACP behavior related to certificate handling and registrar. There, it became clear that the ACP draft should specify generic ACP registrar behavior independent of BRSKI so ACP could be implemented with or without BRSKI and any manual/proprietary or future standardized BRSKI alternatives (for example via NETCONF) would understand the requirements for ACP registrars and its certificate handling.

This lead to additional text about ACP registrars in the ACP document:

1. Defined relationship ACP / ANI (ANI = ACP + BRSKI).

6.1.4 (new) Overview of TA required for ACP.

6.1.5.5 Added explanations/requirements for Re-enrollment.

6.10.7 Normative requirements for ACP registrars (BRSKI or not).

10.2 Operational expectations against ACP registrars (BRSKI or not).

15.1.3. Normative enhancements since start of IESG review

In addition to above ACP registrar / BRSKI related enhancements there is a range of minor normative (also explanatory) enhancements since the start of IESG review:

6.1.1 Hex digits in ACP domain information field now upper-case (no specific reason except that both options are equally good, but capitalized ones are used in rfc5234).

6.1.5.3 Added explanations about CRLs.

6.1.5.6 Added explanations of behavior under failing certificates.

6.1.2 Allow ACP address '0' in ACP domain information field: presence of address indicates permission to build ACP secure channel to node, 0 indicates that the address of the node is assigned by (future) other means than certificate. Non-autonomic nodes have no address at all (that was in -13), and can only connect via ACP connect interfaces to ACP.

6.1.3 Distinction of real ACP nodes (with address) and those with domain certificate without address added as a new rule to ACP domain membership check.

6.6 Added throttling of secure-channel setup attempts.

6.11.1.14 Removed requirement to handle unknown destination ACP traffic in low-end nodes that would never be RPL roots.

6.12.5 Added recommendation to use IPv6 DAD.

6.1.1, 6.7.1.1, 6.7.2, 6.7.3, 6.8.2 Various refined additional certificate, secure channel protocol (IPsec/IKEv2 and DTLS) and ACP GRASP TLS protocol parameter requirements to ensure interoperating implementations (from SEC-AD review).

15.1.4. Explanatory enhancements since start of IESG review

Beyond the functional enhancements from the previous two sections, the majority of changes since -13 are additional explanations from review feedback, textual nits and restructuring - with no functional requirement additions/changes.

1.1 Added "applicability and scope" section with summarized explanations.

2. Added in-band vs. out-of-band management definitions.

6.1.2 (was 6.1.1) expanded explanations of reasoning for elements of the ACP domain information field.

6.1.3 refined explanations of ACP domain membership check and justifications for it.

6.5 Elaborated step-by-step secure channel setup.

6.10 Additional explanations for addressing modes, additional table of addressing formats (thanks MichaelR).

6.10.5 introduced 'F' bit position as a better visual representation in the Vlong address space.

6.11.1.1 extensive overhaul to improve readability of use of RPL (from IESG feedback of non-routing/RPL experts).

6.12.2 Added caution about unconfiguring Data-Plane IPv6 addresses and impact to ACP (limitation of current ACP design, and pointint to more details in 10.2).

10.4 New explanations / summary of configurations for ACP (aka: all config is undesirable and only required for integrating with non-autonomic components, primarily ACP-connect and Registrars).

11. Textually enhanced / better structured security considerations section after IESG security review.

A. (new) Moved all explanations and discussions about futures from section 10 into this new appendix. This text should not be removed because it captures a lot of repeated asked questions in WG and during reviews and from users, and also captures ideas for some likely important followup work. But none of this is relevant to implementing (section 6) and operating (section 10) the ACP.

15.2. draft-ietf-anima-autonomic-control-plane-30

-29 did pass all IESG DISCUSS. This version cleans up remaining comments.

Planned to be removed section Appendix A.6 was moved into new Appendix B.1 to be amended by further A.2, A.3 containing text felt to be unfit for publication in RFC (see below). Added reference to this last draft, and referencing those sections ([ACPDRAFT]).

Final discussion with responsible AD (Eric Vyncke): marked all references to [ACPDRAFT] as to be removed from RFC, as this would be too unconventional. Likewise also [ACPDRAFT] reference itself. Added explanation to appendix B.

Comments from Erik Kline:

2. Fine tuned ULA definition.

Comments Michael Richardson / Eric Vyncke.

6.2.4. / 11. Removed text arguing ability how to use public CA (or not). Replaced with reference to new [ACPDRAFT] section B.3 (not in RFC) that explains current state of understanding (unfinished).

B.3 New text detailing authors understanding of use of public CA (will not be in RFC).

Comments/proposals from Ben Kaduk:

Various: Replaced RFC4492 with RFC8422 which is superceding it.

6.1 Text fix for hash strength 384 bits (from SHA384); Text fix for ec_point_format extension.

6.2.1 Text fixup. Removed requirements for ECDH support in certificate, instead merely explaining the dependencies required IF this is desired (educational).

6.2.5.4. Fine tuning 2 sentences.

6.3.2. (ACP domain membership check) Add reference to ACPDRAFT B.2 explaining why ACP domain membership does not validate ACP address of the connection.

6.4. Downgraded SHOULD to MAY in new -29 suggestion how to deal with DoS attacks with many GRASP announcements. Will also separately ask TSV ADs.

6.4. Fixed extension points in CDDL objective-value definitions (with help from Carsten/Brian).

9.3.5.2. Added explanation when ACP greenfield state ends, and refined text explaining how to deal with this.

11. removed duplicate paragraph (first, kept paragraph was the fixed up, improved correct version).

11. Added references to ACPDRAFT B.1, B.2 as possible future solutions for downgrade attacks.

12. Fixed up text for IANA code point allocation request.

A.6 - removed.

A.9.9 - added one explanatory intro paragraph to makes it easier to distinguish this option from the B.1 considerations.

B.1 - new text suggested from Ben, replacing A.6 (will not be in RFC).

B.2 - new text discussing why there is no network layer address verification in ACP domain membership check (will not be in RFC).

B.4 - Text discussing DULL GRASP attacks via port sweeps and what do do against it.

Other.

1. Added sentence about FCC outage report from June as example for in-band management.

15. added reference to github where document was developed (removed in RFC, part of changelog).

15.3. draft-ietf-anima-autonomic-control-plane-29

Comments from Robert Wilton:

Improved several textual nits.

Discuss/Comments from Erik Kline:

Editorial suggestions and nits. Thanks!.

6.1.3 Added text about how/why rsub is irrelevant for domain membership check.

6.3 Added extension points to AN_ACP DULL GRASP objective because for example ACP domain certificate could be a nice optional additional parameter and prior syntax would have forced us to encode into separate objective unnecessarily.

6.7 Using RFC8415 terminology for exponential backoff parameters.

6.11.2 Amended ACP Sub-Addressing table with future code points, explanations and prefix announced into RPL.

6.12.1.11. Reworked text to better explain how black hole route works and added explanation for prefix for manual address scheme.

8.1.3. Reworked explanation of RIOS for ACP connect interfaces for Type C vs. Type A/B hosts.

8.1.4. Added explanation how this "VRF select" option is required for auto-attachment of Type A/B hosts to ACP and other networks.

Discuss/Comments from Barry Leiba:

Various editorial nits - thanks.

6.1 New section pulling in TLS requirements, no need anymore to duplicat for ACP GRASP, EST, BRSKI (ACP/ANI nodes) and (if desired) OCSP/CRLDP. Added rule to start use secure channel only after negotiation has finished. Added rules not to optimize negotiation across multiple L2 interfaces to the same peer.

6.6 Changed role names in secure channel negotiation process: Alice/Bob -> Decider/Follower. Explanation enhancements. Added definition for ACP nodes with "0" address.

6.8.3 Improved explanation how IKEv2 forces preference of IPsec over GRE due to ACP IPsec profiles being Tunneled vs. Transport.

6.8.4 Limited mentioning of DTLS version requirements to this section.

6.9.2 Removed TLS requirements, they are now in 6.1.

6.10.6 Removed explanation of IANA allocation requirement. Redundant - already in IANA section, and was seen as confusing.

8.1.1 Clarified that there can be security impacts when weakening directly connected address RPF filtering for ACP connect interfaces.

Discuss/Comments from Ben Kaduk:

Many good editorial improvements - thanks!.

5. added explanation of what to do upon failed secure channel establishment.

6.1.1. refined/extended cert public key crypto algo and better distinguished algo for the keys of the cert and the key of the signer.

6.1.1. and following: explicitly defining "serialNumber" to be the X.520 subject name serialNumber, not the certificate serial Number.

6.1.1. emphasize additional authorization step for EST servers (id-kp-cmcRA).

6.1.2 changed AcpNodeName ABNF to again use 32HEXDIG instead of self-defined variation, because authors overlooked that ABNF is case agnostic (which is fine). Added recommendation to encode as lower case. Added full ABNF encoding for extensions (any characters as in "atoms" except the new "+" separator).

6.1.5.3. New text to explain reason for use of HTTPS (instead of HTTP) for CRLDP and when and how to use HTTPS then.

6.1.5.5. added text explaining why/how and when to maintain TA data upon failing cert renewal (one version with BRSKI, one version with other, less secure bootstrap protocols).

6.3. new text and requirement about the signaling of transport ports in DULL GRASP - benefits (no well-known ports required), and problems (additional DoS attack vector, albeit not worse than pre-existing ones, depending on setup of L2 subnets.).

6.7.3.1.1. Specified AUTH_HMAC_SHA2_256_128 (as the ESP authentication algorithm).

6.8.2. Added recommendations for TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256 when supporting TLS 1.3.

8.2.2. Added explanation about downgrade attack across configured ACP tunnels and what to do against it.

9.3.5.2. Rewrote most of section as it originally was too centric on BRSKI. Should now well describe expectations against automated bootstrap. Introduces new requirement not to call node as in support of ANI if is ALSO has TOFU bootstrap.

11. Expanded text about malicious EST servers. Added paragraph about ACP secure channel downgrade attacks. Added paragraphs about private PKI as a core to allow security against fake certificates, added paragraph about considerations/problems when using public PI.

A.10.9 New appendix suggesting how to discover ACP secure channel negotiation downgrade attacks.

Discuss from Roman Danyliw:

6.1.5.1 - Added requirement to only announce SRV.est when a working CA connection.

15 - Amended security considerations with text about registrar dependencies, security of IDevID/ACP-certificate, EST-Server and GRASP for EST server discovery.

Other:

Conversion to XML v3. Solved empty () taxonomy xref problems. Various formatting fixes for v3.

Added contributors section.

15.4. draft-ietf-anima-autonomic-control-plane-28

IESG review Roman Danyliw:

6. Requested additional text elaborating misconfiguration plus attack vectors.

6.1.3.1 Added paragraph about unsecured NTP as basis for time in the absence of other options.

6.7.2 reworded text about additional secure channel protocol requirements.

6.7.3.1.2. Added requirement for ACP nodes supporting IKEv2 to support RFC8247 (not sure how that got dropped from prior versions).

Replaced minimum crypto requirements definition via specific AES options with more generic "symmetric key/hash strength" requirements.

6.10.7.3. Added example how to derive addressing scheme from IDevID (PID). Added explanation how to deal with non-persistent registrar address database (hint: it sucks or is wasteful, what did you expect).

8.1.1. Added explanation for 'Physical controlled/secured'.

8.1.5. Removed 'Physical controlled/secured' text, refer back to 8.1.1.

8.2.1. Fixed ABNF 'or' syntax line.

9.3.2. Added explanation of remote management problem with interface "down" type commands.

10.2.1. Added explanations for attacks from impaired ACP nodes.

11. Rewrote intro paragraph. Removed text referring to enrollment/registrars as they are out of scope of ACP (dependencies only).

11. Added note about need for new protocols inside ACP to use end-to-end authentication.

11. Rewrote paragraph about operator mistakes so as to be actionable. Operators must not make mistakes - but ACP minimizes the mistakes they can make.

ACP domain certificate -> ACP certificate.

Various other cosmetic edits (thanks!) and typo fixes (sorry for not running full spell check for every version. Will definitely do before RFC editor).

Other:

6.12.5.2.1./6.12.5.2.2. Added text explaining link breakage wrt. RTL (came about re-analyzing behavior after question about hop count).

Removed now unnecessary references for earlier rrc822Name otherName choice.

15.5. draft-ietf-anima-autonomic-control-plane-27

Too many revisions with too many fixes. Lets do a one-word change revision for a change now if it helps to accelerate the review process.

Added "subjectAltName /" to make it unambiguous that AcpNodeName is indeed a SAN (from Russ).

15.6. draft-ietf-anima-autonomic-control-plane-26

Russ Housley review of -25.

1.1 Explicit reference for TLS 1.2 RFC.

2. Changed term of "ACP Domain Information" to AcpNodeName (ASN.1) / acp-node-name (ABNF), also through rest of document.

2. Improved CA behavior definition. changed IDevID/LDevID to IDevID/LDevID certificate to be more unambiguous.

2. Changed definition of root CA to just refer to how its used in RFC7030 CA root key update, because thats the only thing relevant to ACP.

6.1.1 Moved ECDH requirement to end of text as it was not related to the subject of the initial paragraphs. Likewise reference to CABFORUM.

6.1.1 Reduced cert key requirements to only be MUST for certs with 2048 RSA public key and P-256 curves. Reduced longer keys to SHOULD.

6.1.2 Changed text for conversion from rfc822Name to otherName / AcpNode, removed all the explanations of benefits coming with rfc822Name *sob* *sob* *sob*.

6.1.2.1 New ASN.1 definition for otherName / AcpNodeName.

6.1.3 Fixed up text. re the handling of missing connectivity for CRLDP / OCSP.

6.1.4 Fixed up text re. inability to use public CA to situation with otherName / AcpNodeName (no more ACME rfc822Name validation for us *sob* *sob* *sob*).

12. Added ASN.1 registration requests to IANA section.

Appenices. Minor changes for rfc822Name to otherName change.

Various minor verbal fixes/enhancements.

15.7. draft-ietf-anima-autonomic-control-plane-25

Crypto parameter discuss from Valery Smyslov and Paul Wouters and resulting changes.

6.7.2 Moved Michael Richardson suggested diagnostic of signaling TA from IPsec section to this general requirements section and added explanation how this may be inappropriate if TA payload is considered secret by TA owner.

6.7.3.1 Added traffic selectors for native IPsec. Improved text explanation.

6.7.3.1.2 removed misleading text about signaling TA when using intermediate certs.

6.7.3.1.2 Removed requirement for 'PKCS #7 wrapped X.509 certificate' requirement on request of Valery Smyslov as it is not defined in RFC7296 and there are enough options mandated in RFC7296. Replaced with just informative text to educate readers who are not IPsec experts what the mandatory option in RFC7296 is that allows to signal certificates.

6.7.3.1.2 Added SHOULD requirement how to deal with CERTREQ so that 6.7.2 requirement for TA diagnostics will work in IKEv2 (ignoring CERTREQ is permitted by IKEv2). Added explanation how this will result in TA cert diagnostics.

6.7.3.1.2 Added requirement for IKEv2 to operate on link-local addresses for ACP so as to assume ACT cert as the only possible authenticator - to avoid potentially failing section from multiple available certs on a router.

6.7.3.1.2 fixed PKIX- style OID to ASN.1 object AlgorithmIdentifier (Paul).

6.7.3.2 Added IPsec traffic selectors for IPsec with GRE.

6.7.5 Added notion that IPsec/GRE MAY be preferred over IPsec/native. Luckily IPsec/native uses tunneling, whereas IPsec/GRE uses transport mode, and there is a long discuss whether it is permitted to even build IPsec connectings that only support transports instead of always being able to fall back to tunnel mode. Added explanatory paragraph why ACP nodes may prefer GRE over native (wonder how that was missing..).

9.1.1 Added section to explain need for secure channel peer diagnostics via signaling of TA. Four examples given.

Paul Wouters mentioned that ipkcs7 had to be used in some interop cases with windows CA, but that is an issue of ACP Registrar having to convert into PKCS#7 to talk to a windows CA, and this spec is not concerned with that, except to know that it is feasible, so not mentioned in text anywhere, just tracking discussion here in changelog.

Michael Richardson:

3.1.3 Added point in support of rfc822address that CA may not support to sign certificates with new attributes (such as new otherName).

Michael Richardson/Brian Carpenter fix:

6.1.5.1/6.3 Fixed GRASP examples.

Joe Halpern review:

1. Enhanced introduction text for in-band and of out-of-band, explaining how ACP is an in-band network aiming to achieve all possible benefits of an out-of-band network.

1. Comprehensive explanation for term Data-Plane as it is only logically following pre-established terminology on a fully autonomic node, when used for existing nodes augmented with ACP, Data-Plane has more functionality than usually associated with the term.

2. Removed explanatory text for Data-Plane, referring to section 1.

2. Reduced explanation in definition of in-band (management/signaling), out-of-band-signaling, now pointing to section 1.

5. Rewrote a lot of the steps (overview) as this text was not reviewed for long time. Added references to normative section for each step to hopefully avoid feedback of not explaining terms used (really not possible to give good summary without using forward references).

2. Separate out-of-band-management definition from virtual out-of-band-management definition (later one for ACP).

2. Added definitions for RPI and RPL.

6.1.1. added note about end-to-end authentication to distinguish channel security from overall ACP security model.

6.5 Fixed bugs in channel selection signaling step description (Alice vs. Bob).

6.7.1 Removed redundant channel selection explanation.

6.10.3 remove locator/identifier terminology from zone addressing scheme description (unnecessary), removed explanations (now in 9.4), simplified text, clarified requirement for Node-ID to be unique, recommend to use primarily zone 0.

6.10.3.1 Removed. Included a lot of insufficient suggestions for future standards extensions, most of it was wrong or would need to be revisited by WG anyhow. Idea now (just here for comment): Announce via GRASP Zone-ID (e.g. from per-zone edge-node/registrar) into a zone of the ACP so all nodes supporting the scheme can automatically self-allocate the Zone-ID.

6.11.1.1 (RPL overview), eliminated redundant text.

6.11.1.1.1 New subsection to better structure overview.

6.11.1.1.2 New subsection to better group overview, replaced TTL explanation (just the symptom) with hopefully better reconvergence text (intent of the profile) for the ACP RPL profile.

6.11.1.1.6 Added text to explain simple choice for rank_factor.

6.11.1.1.13 moved explanation for RPI up into 6.11.1.1.

6.12.5.1 rewrote section for ACP Loopback Interface.

9.4 New informative/informational section for partial or incremental adoption of ACP to help understand why there is the Zone interface sub-scheme, and how to use it.

Unrelated fixes:

Ask to RFC editor to add most important abbreviations to RFC editor abbreviation list.

6.10.2 changed names in ACP addressing scheme table to be less suggestive of use.

Russ Hously review:

2. Fixed definition of "Enrollment", "Trust Anchor", "CA", and "root CA". Changed "Certificate Authority" to "Certification Authority" throughout the document (correct term according to X.509).

6.1 Fixed explanation of mutual ACP trust.

6.1.1 s/X509/X509v3/.

6.1.2 created bulleted lists for explanations and justifications for choices of ACP certificate encoding. No semantic changes, just to make it easier to refer to the points in discussions (rfcdiff seems to have a bug showing text differences due to formatting changes).

6.1.3 Moved content of rule #1 into previous rule #2 because certification chain validation does imply validation of lifetime. numbers of all rules reduced by 1, changed hopefully all references to the rule numbers in the document.

Rule #3, Hopefully fixed linguistic problem self-contradiction of MUST by lower casing MUST in the explanation part and rewriting the condition when this is not applicable.

6.1.4 Replaced redundant term "Trust Point" (TP) with Trust Anchor (TA). Replaced throughout document Trust Anchor with abbreviation TA.

Enhanced several sentences/rewrote paragraphs to make explanations clearer.

6.6 Added explanation how ACP nodes must throttle their attempts for connection making purely on the result of their own connection attempts, not based on those connections where they are responder.

15.8. draft-ietf-anima-autonomic-control-plane-24

Leftover from -23 review by Eric Vyncke:

Swapping sections 9 and 10, section 9 was meant to be at end of document and summarize. Its not meant to be misinterpreted as introducing any new information. This did happen because section 10 was added after section 9.

15.9. draft-ietf-anima-autonomic-control-plane-23

Note: big rfcdiff of TOC is an rfcdiff bug, changes really minimal.

Review of IPsec security with Mcr and ipsec mailing list.

6.7.1 - new section: Moved general considerations for secure channel protocols here, refined them.

6.7.2 - new section: Moved common requirements for secure channel protocols here, refined them.

6.7.3.1.1. - improved requirements text related to RFC8221, better explanations re. HW acceleration issues.

6.7.3.1.2. - improved requirements text related to RFC8247, (some requirements still discussed to be redundant, will be finalized in next weeks.

Eric Vyncke review of -21:

Only noting most important changes, long list of smaller text/readability enhancements.

2. - New explanation of "normative", "informational" section title tags. alphabetic reordering of terms, refined definitions for CA, CRL. root CA.

6.1.1. - explanation when IDevID parameters may be copied into LDevID.

6.1.2. - Fixed hex digits in ACP domain information to lower case.

6.1.3.1. - New section on Realtime clock and Time Validation.

6.3 - Added explanation that DTLS means >= version 1.2 (not only 1.2).

6.7 - New text in this main section explaing relationship of ACP secure channels and ACP virtual interfaces - with forward references to virtual interface section.

6.8.2 - reordered text and picture, no text change.

6.10.7.2 - describe first how Registrar-ID can be allocated for all type of registrars, then refined text for how to potentially use MAC addresses on physical registrars.

6.11.1.1 - Added text how this profile does not use Data-Plane artefacts (RPI) because hardware forwarding. This was previously hidden only later in the text.

6.11.1.13. - Rewrote RPL Data-Plane artefact text. Provide decoder ring for abbreviations and all relevant RFCs.

6.12.5.2. - Added more explicit text that secure channels are mapped into virtual interfaces, moved different type of interfaces used by ACP into separate subsections to be able to refer to them.

7.2 - Rewrote/refined text for ACP on L2, prior text was confusing and did not well explain why ACP for L2/L3 switches can be implemented without any L2 (HW) changes. Also missing explanation of only running GRASP untagged when VLANs are used.

8.1.1 - Added requirement for ACP Edge nodes to allow configurable filtering of IPv6 RPI headers.

11. - (security section). Moved explanation of address stealing from 7.2 to here.

15.10. draft-ietf-anima-autonomic-control-plane-22

Ben Kaduk review of -21:

RFC822 encoding of ACP domain information:

6.1.2 rewrote text for explaining / justifying use of rfc822name as identifier for node CP in certificate (was discussed in thread, but badly written in prior versions).

6.1.2 Changed EBNF syntax to use "+" after rfcSELF because that is the known primary name to extensions separator in many email systems ("." was wrong in prior versions).

6.1.2 Rewrote/improved explanations for use of rfc822name field to explain better why it is PKIX compliant and the right thing to do.

Crypto parameters for IPsec:

6.1 - Added explanation of why manual keying for ACP is not feasible for ACP. Surprisingly, that text did not exist. Referred to by IPsec text (6.7.1), but here is the right place to describe the reasoning.

6.1.2 - Small textual refinement referring to requirements to authenticate peers (for the special cases of empty or '0' ACP address in ACP domain information field).

6.3 - To better justify Bens proposed change of secure channel protocol being IPsec vs. GRASP objective being IKEv2, better explained how protocol indicated in GRASP objective-value is name of protocol used to negotiate secure channel, use example of IKEv2 to negotiate IPsec.

6.7.1 - refinemenet similar to 6.3.

- moved new paragraph from Bens pull request up from 6.7.1.1 to 6.7.1 as it equally applies to GRE encapped IPsec (looks nicer one level up).

- created subsections 6.7.1.1 (IPsec/ESP) / 6.7.1.2 (IKEv2) to clearer distinguish between these two requirements blocks.

- Refined the text in these two sections to hopefully be a good answer to Valery's concern of not randomly mocking with existing requirements docs (rfc8247 / rfc8221).

6.7.1.1.1 - IPsec/ESP requirements section:

- MUST support rfc8221 mandatory EXCEPT for the superceeding requirements in this section. Previously, this was not quite clear from the text.

- Hopefully persuasive explanations about the requirements levels for ENCR_AES_GCM_16, ENCR_AES_CBC, ENCR_AES_CCM_8 and ENCR_CHACHA20_POLY1305: Restructured text for why not ENCR_AES_CBC (was in prior version, just not well structured), added new explanations for ENCR_AES_CCM_8 and ENCR_CHACHA20_POLY130.

- In simple terms, requirements for ENCR_AES_CBC, ENCR_AES_CCM_8, ENCR_CHACHA20 are SHOULD when they are implementable with equal or faster performance than ENCR_AES_GCM_16.

- Removed text about "additional rfc8221" requirements MAY be used. Now the logic is that all other requirements apply. Hopefully we have written enough so that we prohibited downgrades.

6.7.1.1.2 - RFC8247 requirements:

- Added mandate to support rfc8247, added explanation that there is no "stripping down" requirement, just additional stronger requirements to mandate correct use of ACP certificates during authentication.

- refined text on identifying ACP by IPv6 address to be clearer: Identifying in the context of IKEv2 and cases for '0' in ACP domain information.

- removed last two paragraphs about relationship to rfc8247, as this is now written in first paragraph of the section.

End of Ben Kaduk review related fixes.

Other:

Forgot to update example of ACP domain information to use capitalized hex-digits as required by HEXDIG used.

Added reference to RFC8316 (AN use-cases) to beginning of section 3 (ACP use cases).

Small Enhanced IPsec parameters description / requirements fixes (from Michael Richardson).

16. Normative References

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

Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-bootstrapping-keyinfra-43, 7 August 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-bootstrapping-keyinfra-43.txt>>.

[I-D.ietf-anima-grasp]

Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-grasp-15.txt>>.

[IKEV2IANA]

IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <<https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml>>.

- [RFC1034] Mockapetris, P.V., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, DOI 10.17487/RFC6551, March 2012, <<https://www.rfc-editor.org/info/rfc6551>>.
- [RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <<https://www.rfc-editor.org/info/rfc6552>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<https://www.rfc-editor.org/info/rfc6553>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.

- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7676] Pignataro, C., Bonica, R., and S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC 7676, DOI 10.17487/RFC7676, October 2015, <<https://www.rfc-editor.org/info/rfc7676>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.
- [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <<https://www.rfc-editor.org/info/rfc8247>>.
- [RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <<https://www.rfc-editor.org/info/rfc8422>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

17. Informative References

- [ACPDRAFT] Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-30, <<https://tools.ietf.org/html/draft-ietf-anima-autonomic-control-plane-30.pdf>>. [RFC-Editor: Please remove this complete reference from the RFC] Refer to the IETF working group draft for the few sections removed from this document for various reasons. They capture the state of discussion about unresolved issues that may need to be revisited in future work.
- [AR8021] Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [CABFORUM] CA/Browser Forum, "Certificate Contents for Baseline SSL", November 2019, <<https://cabforum.org/baseline-requirements-certificate-contents/>>.
- [FCC] FCC, "FCC STAFF REPORT ON NATIONWIDE T-MOBILE NETWORK OUTAGE ON JUNE 15, 2020 (PS Docket No. 20-183)", 2020, <<https://docs.fcc.gov/public/attachments/DOC-367699A1.docx>>. The FCC's Public Safety and Homeland Security Bureau issues a report on a nationwide T-Mobile outage that occurred on June 15, 2020. Action by: Public Safety and Homeland Security Bureau.
- [I-D.eckert-anima-noc-autoconfig]
Eckert, T., "Autoconfiguration of NOC services in ACP networks via GRASP", Work in Progress, Internet-Draft, draft-eckert-anima-noc-autoconfig-00, 2 July 2018, <<http://www.ietf.org/internet-drafts/draft-eckert-anima-noc-autoconfig-00.txt>>.
- [I-D.ietf-acme-star]
Sheffer, Y., Lopez, D., Dios, O., Pastor, A., and T. Fossati, "Support for Short-Term, Automatically-Renewed (STAR) Certificates in Automated Certificate Management Environment (ACME)", Work in Progress, Internet-Draft, draft-ietf-acme-star-11, 24 October 2019, <<http://www.ietf.org/internet-drafts/draft-ietf-acme-star-11.txt>>.
- [I-D.ietf-anima-prefix-management]
Jiang, S., Du, Z., Carpenter, B., and Q. Sun, "Autonomic IPv6 Edge Prefix Management in Large-scale Networks", Work in Progress, Internet-Draft, draft-ietf-anima-prefix-

management-07, 18 December 2017, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-prefix-management-07.txt>>.

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

Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", Work in Progress, Internet-Draft, draft-ietf-anima-reference-model-10, 22 November 2018, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-reference-model-10.txt>>.

[I-D.ietf-roll-applicability-template]

Richardson, M., "ROLL Applicability Statement Template", Work in Progress, Internet-Draft, draft-ietf-roll-applicability-template-09, 3 May 2016, <<http://www.ietf.org/internet-drafts/draft-ietf-roll-applicability-template-09.txt>>.

[I-D.ietf-tls-dtls13]

Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-38, 29 May 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-tls-dtls13-38.txt>>.

[IEEE-1588-2008]

IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", December 2008, <<http://standards.ieee.org/findstds/standard/1588-2008.html>>.

[IEEE-802.1X]

Group, W. -. H. L. L. P. W., "IEEE Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control", February 2010, <<http://standards.ieee.org/findstds/standard/802.1X-2010.html>>.

[LLDP]

Group, W. -. H. L. L. P. W., "IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery", June 2016, <<https://standards.ieee.org/findstds/standard/802.1AB-2016.html>>.

- [MACSEC] Group, W. - . H. L. L. P. W., "IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security", June 2006, <<https://standards.ieee.org/findstds/standard/802.1AE-2006.html>>.
- [RFC1112] Deering, S.E., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC1492] Finseth, C., "An Access Control Protocol, Sometimes Called TACACS", RFC 1492, DOI 10.17487/RFC1492, July 1993, <<https://www.rfc-editor.org/info/rfc1492>>.
- [RFC1654] Rekhter, Y., Ed. and T. Li, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 1654, DOI 10.17487/RFC1654, July 1994, <<https://www.rfc-editor.org/info/rfc1654>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <<https://www.rfc-editor.org/info/rfc2315>>.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.
- [RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, DOI 10.17487/RFC3164, August 2001, <<https://www.rfc-editor.org/info/rfc3164>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.

- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <<https://www.rfc-editor.org/info/rfc3411>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <<https://www.rfc-editor.org/info/rfc3596>>.
- [RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, DOI 10.17487/RFC3920, October 2004, <<https://www.rfc-editor.org/info/rfc3920>>.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004, <<https://www.rfc-editor.org/info/rfc3954>>.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <<https://www.rfc-editor.org/info/rfc4007>>.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <<https://www.rfc-editor.org/info/rfc4429>>.
- [RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <<https://www.rfc-editor.org/info/rfc4541>>.

- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <<https://www.rfc-editor.org/info/rfc4604>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC4610] Farinacci, D. and Y. Cai, "Anycast-RP Using Protocol Independent Multicast (PIM)", RFC 4610, DOI 10.17487/RFC4610, August 2006, <<https://www.rfc-editor.org/info/rfc4610>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", RFC 4985, DOI 10.17487/RFC4985, August 2007, <<https://www.rfc-editor.org/info/rfc4985>>.
- [RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, DOI 10.17487/RFC5790, February 2010, <<https://www.rfc-editor.org/info/rfc5790>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <<https://www.rfc-editor.org/info/rfc5912>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6402] Schaad, J., "Certificate Management over CMS (CMC) Updates", RFC 6402, DOI 10.17487/RFC6402, November 2011, <<https://www.rfc-editor.org/info/rfc6402>>.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<https://www.rfc-editor.org/info/rfc6554>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <<https://www.rfc-editor.org/info/rfc6733>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/info/rfc6824>>.

- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7404] Behringer, M. and E. Vyncke, "Using Only Link-Local Addressing inside an IPv6 Network", RFC 7404, DOI 10.17487/RFC7404, November 2014, <<https://www.rfc-editor.org/info/rfc7404>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.
- [RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, June 2015, <<https://www.rfc-editor.org/info/rfc7576>>.
- [RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <<https://www.rfc-editor.org/info/rfc7585>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.

- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8316] Nobre, J., Granville, L., Clemm, A., and A. Gonzalez Prieto, "Autonomic Networking Use Case for Distributed Detection of Service Level Agreement (SLA) Violations", RFC 8316, DOI 10.17487/RFC8316, February 2018, <<https://www.rfc-editor.org/info/rfc8316>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <<https://www.rfc-editor.org/info/rfc8368>>.
- [RFC8398] Melnikov, A., Ed. and W. Chuang, Ed., "Internationalized Email Addresses in X.509 Certificates", RFC 8398, DOI 10.17487/RFC8398, May 2018, <<https://www.rfc-editor.org/info/rfc8398>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

- [RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <<https://www.rfc-editor.org/info/rfc8572>>.
- [X.509] International Telecommunication Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, ISO/IEC 9594-8, October 2016, <<https://www.itu.int/rec/T-REC-X.509>>.
- [X.520] International Telecommunication Union, "Information technology - Open Systems Interconnection - The Directory: Selected attribute types", ITU-T Recommendation X.520, ISO/IEC 9594-6, October 2016, <<https://www.itu.int/rec/T-REC-X.520>>.

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 6.11.7. 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 via separate prefixes.

In networks that always expect to rely on a centralized PMS as described above (Section 9.2.5), 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 a new otherName was introduced, 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-node-name.

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 certificates.

A.2. BRSKI Bootstrap (ANI)

BRSKI describes how nodes with an IDevID certificate can securely and zero-touch enroll with an LDevID certificate 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 certificate 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 otherName / AcpNodeName 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 Certification Authority in an ACP network must not change the otherName / AcpNodeName 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 certificate 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 ([RFC8572]) 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 DNS (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 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 DODAG, which are loop-free distribution trees, to make GRASP flooding more efficient both for M_FLOOD and M_DISCOVERY. See Section 6.13.5 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. CAs, domains and routing subdomains

There is a wide range of setting up different ACP solution by appropriately using CAs and the domain and rsub elements in the acp-node-name in 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 that can authenticate each other as belonging to the same ACP network using the ACP domain membership check (Section 6.2.3). GRASP inside the ACP is run across all transitively connected ACP nodes in a domain.

The rsub element in the acp-node-name 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 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 that would be required if other routing protocols were 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 acp-node-name. 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 acp-node-name 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.7. 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. Its applicability for use is quite flexible and freeform, with potential applications including 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 TA and domain names to the parameters for the ACP domain membership check (Section 6.2.3) 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.8. 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 omitted. 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 the Decider and who is the Follower 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 RPL used for the ACP 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.9. Further (future) options

A.9.1. 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 acp-node-name) 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.9.2. More options for avoiding IPv6 Data-Plane dependencies

As described in Section 6.13.2, 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) 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.9.3. 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 6.3) and ACP GRASP need to be included into such model/API.

A.9.4. RPL enhancements

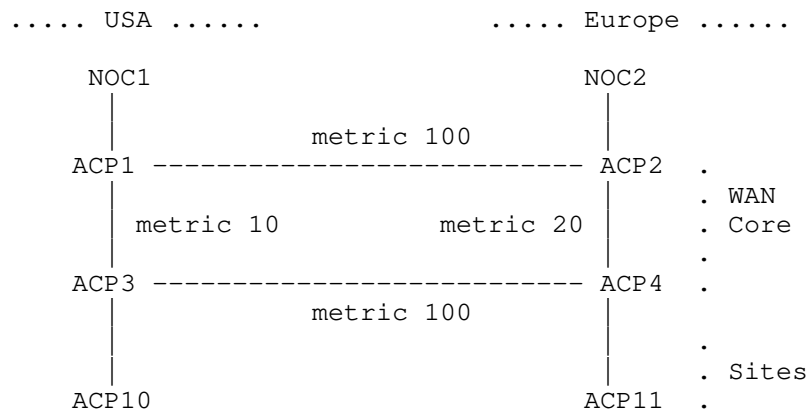


Figure 19: Dual NOC

The profile for RPL specified in this document builds only one spanning-tree path set to a root, typically a registrar in one NOC. In the presence of multiple NOCs, routing toward the non-root NOCs may be suboptimal. Figure 19 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.9.5. 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 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.9.6. 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.9.7. Diagnostics

Section 9.1 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 (noting that such information is conveyed in an insecure manner). 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 certificate of BRSKI pledges should be included in the selected insecure diagnostics option. This may be undesirable when exposure of device information is seen as too much of a security issue (ability to deduce possible attack vectors from device model for example).
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.9.8. 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 cannot 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 through 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 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.

A.9.9. Detecting ACP secure channel downgrade attacks

The following text proposes a mechanism to protect against downgrade attacks without introducing a new specialized UPFRONT GRASP secure channel mechanism. Instead, it relies on running GRASP after establishing a secure channel protocol to verify if the established secure channel option could have been the result of a MITM downgrade attack:

MITM attackers can force downgrade attacks for ACP secure channel selection by filtering/modifying DULL GRASP messages and/or actual secure channel data packets. For example, if at some point in time DTLS traffic could be easier decrypted than traffic of IKEv2, the MITM could filter all IKEv2 packets to force ACP nodes to use DTLS (assuming the ACP nodes in question supported both DTLS and IKEv2).

For cases where such MITM attacks are not capable to inject malicious traffic (but only to decrypt the traffic), a downgrade attack could be discovered after a secure channel connection is established, for example by use of the following type of mechanism:

After the secure channel connection is established, the two ACP peers negotiate via an appropriate (To Be Defined) GRASP negotiation which ACP secure channel protocol should have been selected between them (in the absence of a MITM attacker). This negotiation would have to

signal the DULL GRASP announced ACP secure channel options by each peer followed by an announcement of the preferred secure channel protocol by the ACP peer that is the Decider in the secure channel setup, e.g. the ACP peer that is deciding which secure channel protocol to pick. If that chosen secure channel protocol is different from the one that actually was chosen, then this mismatch is an indication that there is a MITM attacker or other similar issue (firewall prohibiting the use of specific protocols) that caused a non-preferred secure channel protocol to be chosen. This discovery could then result in mitigation options such as logging and ensuing investigations.

Appendix B. Unfinished considerations (To Be Removed From RFC)

[RFC-Editor: This whole appendix B. and its subsections to be removed for the RFC.

This appendix contains unfinished considerations that are removed from the RFC, they are maintained in this draft as a log of the state of discussion and point of reference. Together with this appendix, also the references pointing to it are marked to be removed from the RFC because no consensus could be reached that a self-reference to a draft version of the RFC is an appropriate breadcrumb to point to unfinished considerations.

The authors plan to move these considerations into a new target informational draft, please look for draft-eckert-anima-acp-considerations.

B.1. Considerations for improving secure channel negotiation

Proposed text from Benjamin Kaduk. It is suggested to replace the text of appendix A.6 in previous versions of this draft (up to version 29).

The discovery procedure in this specification for low-level ACP channel support by layer-2 peers involves DULL GRASP and attempting (usually in parallel) to establish all supported channel types, learning the peer ACP address and correspondingly the assignment of Decider and Follower roles, and tearing down all channels other than the one preferred by the Decider. This procedure, in general, becomes resource intensive as the number of possible secure channels grows; even worse, under some threat models, the security of the discovery result is only as strong as the weakest supported secure channel protocol. Furthermore, the unilateral determination of "best" channel type by the Decider does not result in the optimal outcome in all possible scenarios.

This situation is tolerable at present, with only two secure channels (DTLS and IPsec) defined, but long-term agility in the vein of [BCP201] will require the introduction of an alternate discovery/negotiation procedure. While IKEv2 is the IETF standard protocol for negotiating security associations, it currently does not have a defined mechanism flexible enough to negotiate the parameters needed for, e.g., an ACP DTLS channel, let alone for allowing ACP peers to indicate their preference metrics for channel selection. Such a mechanism or mechanisms could be defined, but if ACP agility requires introducing a new channel type, for example MacSec, IKEv2 would again need to be extended in order to negotiate an ACP MacSec association. Making ACP channel agility dependent on updates to IKEv2 is likely to result in obstacles due to different timescales of evolution, since IKEv2 implementations help form the core of Internet-scale security infrastructure and must accordingly be robust and thoroughly tested.

Accordingly, a dedicated ACP channel negotiation mechanism is appropriate as a way to provide long-term algorithm and secure-channel protocol agility. Such a mechanism is not currently defined, but one possible design is as follows. A new DULL GRASP objective is defined to indicate the GRASP-over-TLS channel, which is by definition preferred to other channel types (including DTLS and IPsec). When both peers advertise support for GRASP-over-TLS, GRASP-over-TLS must run to completion before other channel types are attempted. The GRASP-over-TLS channel performs the necessary negotiation by establishing a TLS connection between the peers and using that connection to secure a dedicated GRASP instance for negotiating supported channel types and preference metrics. This provides a rich language for determining what secure channel protocol to use for the ACP link while taking into account the capabilities and preferences of the ACP peers, all protected by the security of the TLS channel.

B.2. ACP address verification

The AcpNodeName of most ACP nodes contains in the acp-address field the primary ACP address to be used by the node for end-to-end connections across ACP secure channels. Nevertheless, there is no verification of an ACP peers address specified in this document. This section explains the current understanding as to why this is not done.

Not all ACP nodes will have an actual IPv6 address in the acp-address field of their AcpNodeName. Those who do not include nodes that do not support ACP secure channels, such as pre-existing NOC equipment that only connects to the ACP via ACP connect interfaces. Likewise, future ACP node type that may want to have their Node-ID not be defined by an ACP registrar, but differently cannot have the ACP

address be provided in their ACP certificate where it would be defined by the registrar. In result, any scheme that would rely on verification of the acp-address in the ACP certificate would only apply to a subset of ACP nodes.

The transport stack network layer address used for ACP secure channels is not the acp-address. For automatically established ACP secure channels, it is a link-local IPv6 address. For explicitly configured ACP secure channels (to reach across non ACP L3 network segments), the address is any IPv4 or IPv6 address routable to that remote destination.

When the acp-address is actually used across the ACP, it can only be verified by a peer when the peer has the certificate of the peer. Unless further higher layer mechanisms are developed on top of the ACP (for example via ASA), the only mechanism to access a peers ACP certificate is for secure connections in which the peers certificates are exchanged and cryptographically verified, e.g. TLS and DTLS. Initially, it is expected that the ACP will carry many legacy network management control connections that unfortunately not end-to-end authenticated but that are solely protected by being carried across the ACP secure channels. ACP address verification therefore cannot be used for such connections without additional higher layer components.

For the remaining (TLS/DTLS) connections for which address verification can be used, the main question is: what additional benefit would address verification provide?

The main value that transport stack network layer address verification could provide for these type of connections would be the discovery of on-path transport proxies. For example, in case of BRSKI, pledges connect to an ACP registrar via an ASA implementing a TCP proxy because the pledge itself has at that point in time no ACP certificate valid to build ACP secure channels and hence needs to rely on such a proxy. This is one example where such a TCP proxy is required and not a form of attack.

In general, on path TCP proxies could be a form of attack, but it stands to reason, that an attacker that manages to enable a malicious TCP proxy could likely equally build a transparent proxy not changing the network layer addresses. Only when the attacker operates off-path would this option not be possible. Such attacks could indeed be possible: An impaired ACP node could announce itself as another service instance for a service whose utilization it wants to attack. It could then attempt to look like a valid server by simply TCP proxying the clients connections to a valid server and then attack the connections passing through it (passive decrypting or passive

fingerprint analysis). But like the BRSKI proxy, this behavior could be perfectly legitimate and not an attack. For example, TCP has in the past often suffered from performance issues across difficult (high capacity, high loss) paths, and TCP proxies where and are being used simply as a tool for isolating such path segments (such as a WAN), and providing caching and local-retransmit of in-transit packets, reducing the effective path segment capacity.

As explained elsewhere in this document already, considerations for these type of attack are therefore outside the scope of the ACP but fundamental to further design of the ASA infrastructure. Beyond distinguishing whether a TCP proxy would be beneficial or malicious, the even more fundamental question is how to determine from a multitude of service announcements which instance is the most trustworthy and functionally best. In the Internet/web, this question is NOT solved inside the network but through off-net human interaction ("trust me, the best search engine is www.<insert-your-personal-recommendation>.com").

B.3. Public CA considerations

Public CAs are outside the scope of this document for the following reasons. This appendix describes the current state of understanding for those interested to consider utilizing public CA for the ACP in the future.

If public CA where to be used to enroll ACP nodes and act as TA, this would require a model in which the public CA would be able to assert the ownership of the information requested in the certificate, especially the AcpNodeName, for example mitigated by the domain registrar(s). Due to the use of a new, ACP unique encoding of the AcpNodeName, there is no mechanism for public CA to do so. More importantly though, isolation between ACPs of disjoint operated ACPs is achieved in the current ACP design through disjoint TA. A public CA is in general based on a single (set of) TA shared across all certificates signed by the CA.

Due to the fact that the ACP domain membership check also validates that a peers domain name in the AcpNodeName matches that of the ACP node itself, it would be possible to use the same TA across disjoint ACP domains, but the security and attack implications of such an approach are beyond the scope of this document.

The use of ULA addresses in the AcpNodeName is another novel aspect for certificates from a possible public CA. Typically, ULA addresses are not meant to be signed by a public CA when carried in an address field, because there is no ownership of a particular ULA address in the scope of the Internet, which is what public CA operate on.

Nevertheless, the ULA addresses used by the ACP are scoped to be valid only within the confines of a specific ACP as defined by the domain name in the `AcpNodeName`. However, this understanding has not been reviewed or accepted by any bodies responsible for policies of public CA.

Because in this specification, ACPs are isolated from each other primarily by their TA, when a public CA would intend to sign ACP certificates and using a single TA to sign TA of ACP certificates from different operators/domain, it could do so by ensuring that the domain name in the `AcpNodeName` was a globally owned DNS ACP domain name of the organization, and beyond that, it would need to validate that the ACP registrar of that domain who is mitigating the enrollment is authorized to vouch for the ownership of the `acp-` address within the scope of the ACP domain name.

B.4. Hardening DULL GRASP considerations

DULL GRASP suffers from similar type of DoS attacks as many link-local multicast discovery protocols, for example mDNS. Attackers on a subnet may be able to inject malicious DULL GRASP messages that are indistinguishable from non-malicious DULL GRASP messages to create Denial-of-Service (DoS) attacks that force ACP nodes to attempt many unsuccessful ACP secure channel connections.

When an ACP node sees multiple `AN_ACP` objectives for the same secure channel protocol on different transport addresses, it could prefer connecting via the well-known transport address if the secure channel method has one, such as UDP port 500 for IKEv2. For protocols such as (ACP secure channel over) DTLS for which there are no well defined port number, this heuristic does not provide benefits though.

DoS attack with port numbers can also be eliminated by relying on well known-port numbers implied by the GRASP method-name. For example, a future service name of "DTLSacp" could be defined to be associated only to a newly to be assigned well known UDP port for ACP over DTLS, and the port number in the GRASP transport address information would be ignored. Note that there is already a variety of ports assigned to specific protocols over DTLS by IANA, so especially for DTLS this would not be uncommon.

Authors' Addresses

Toerless Eckert (editor)
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara, 95050
United States of America

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

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: 15 May 2021

M. Pritikin
Cisco
M. Richardson
Sandelman
T.T.E. Eckert
Futurewei USA
M.H. Behringer

K.W. Watsen
Watsen Networks
11 November 2020

Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-45

Abstract

This document specifies automated bootstrapping of an Autonomic Control Plane. To do this a Secure Key Infrastructure is bootstrapped. This is done using manufacturer-installed X.509 certificates, in combination with a manufacturer's authorizing service, both online and offline. We call this process the Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol. 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 deployment models with less stringent security requirements is included. Bootstrapping is complete when the cryptographic identity of the new key infrastructure is successfully deployed to the device. 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 15 May 2021.

Copyright Notice

Copyright (c) 2020 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	5
1.1. Prior Bootstrapping Approaches	6
1.2. Terminology	8
1.3. Scope of solution	11
1.3.1. Support environment	11
1.3.2. Constrained environments	11
1.3.3. Network Access Controls	12
1.3.4. Bootstrapping is not Booting	12
1.4. Leveraging the new key infrastructure / next steps . . .	12
1.5. Requirements for Autonomic Network Infrastructure (ANI) devices	13
2. Architectural Overview	13
2.1. Behavior of a Pledge	15
2.2. Secure Imprinting using Vouchers	16
2.3. Initial Device Identifier	17
2.3.1. Identification of the Pledge	18
2.3.2. MASA URI extension	19
2.4. Protocol Flow	20
2.5. Architectural Components	23
2.5.1. Pledge	23
2.5.2. Join Proxy	23
2.5.3. Domain Registrar	23
2.5.4. Manufacturer Service	23
2.5.5. Public Key Infrastructure (PKI)	24
2.6. Certificate Time Validation	24
2.6.1. Lack of realtime clock	24
2.6.2. Infinite Lifetime of IDevID	24
2.7. Cloud Registrar	25
2.8. Determining the MASA to contact	25
3. Voucher-Request artifact	26

3.1.	Nonceless Voucher Requests	27
3.2.	Tree Diagram	27
3.3.	Examples	27
3.4.	YANG Module	29
4.	Proxying details (Pledge - Proxy - Registrar)	33
4.1.	Pledge discovery of Proxy	34
4.1.1.	Proxy GRASP announcements	35
4.2.	CoAP connection to Registrar	37
4.3.	Proxy discovery and communication of Registrar	37
5.	Protocol Details (Pledge - Registrar - MASA)	38
5.1.	BRSKI-EST TLS establishment details	40
5.2.	Pledge Requests Voucher from the Registrar	41
5.3.	Registrar Authorization of Pledge	43
5.4.	BRSKI-MASA TLS establishment details	43
5.4.1.	MASA authentication of customer Registrar	44
5.5.	Registrar Requests Voucher from MASA	45
5.5.1.	MASA renewal of expired vouchers	47
5.5.2.	MASA pinning of registrar	48
5.5.3.	MASA checking of voucher request signature	48
5.5.4.	MASA verification of domain registrar	49
5.5.5.	MASA verification of pledge prior-signed-voucher-request	50
5.5.6.	MASA nonce handling	50
5.6.	MASA and Registrar Voucher Response	50
5.6.1.	Pledge voucher verification	53
5.6.2.	Pledge authentication of provisional TLS connection	54
5.7.	Pledge BRSKI Status Telemetry	55
5.8.	Registrar audit-log request	56
5.8.1.	MASA audit log response	58
5.8.2.	Calculation of domainID	60
5.8.3.	Registrar audit log verification	61
5.9.	EST Integration for PKI bootstrapping	62
5.9.1.	EST Distribution of CA Certificates	63
5.9.2.	EST CSR Attributes	63
5.9.3.	EST Client Certificate Request	64
5.9.4.	Enrollment Status Telemetry	64
5.9.5.	Multiple certificates	65
5.9.6.	EST over CoAP	66
6.	Clarification of transfer-encoding	66
7.	Reduced security operational modes	66
7.1.	Trust Model	66
7.2.	Pledge security reductions	67
7.3.	Registrar security reductions	68
7.4.	MASA security reductions	69
7.4.1.	Issuing Nonceless vouchers	69
7.4.2.	Trusting Owners on First Use	70
7.4.3.	Updating or extending voucher trust anchors	71

8.	IANA Considerations	72
8.1.	The IETF XML Registry	72
8.2.	YANG Module Names Registry	72
8.3.	BRSKI well-known considerations	72
8.3.1.	BRSKI .well-known registration	72
8.3.2.	BRSKI .well-known registry	73
8.4.	PKIX Registry	73
8.5.	Pledge BRSKI Status Telemetry	73
8.6.	DNS Service Names	74
8.7.	GRASP Objective Names	74
9.	Applicability to the Autonomic Control Plane (ACP)	74
9.1.	Operational Requirements	76
9.1.1.	MASA Operational Requirements	76
9.1.2.	Domain Owner Operational Requirements	77
9.1.3.	Device Operational Requirements	77
10.	Privacy Considerations	78
10.1.	MASA audit log	78
10.2.	What BRSKI-EST reveals	78
10.3.	What BRSKI-MASA reveals to the manufacturer	79
10.4.	Manufacturers and Used or Stolen Equipment	81
10.5.	Manufacturers and Grey market equipment	83
10.6.	Some mitigations for meddling by manufacturers	83
10.7.	Death of a manufacturer	84
11.	Security Considerations	85
11.1.	Denial of Service (DoS) against MASA	86
11.2.	DomainID must be resistant to second-preimage attacks	86
11.3.	Availability of good random numbers	87
11.4.	Freshness in Voucher-Requests	87
11.5.	Trusting manufacturers	88
11.6.	Manufacturer Maintenance of trust anchors	89
11.6.1.	Compromise of Manufacturer IDevID signing keys	91
11.6.2.	Compromise of MASA signing keys	91
11.6.3.	Compromise of MASA web service	93
11.7.	YANG Module Security Considerations	94
12.	Acknowledgements	94
13.	References	94
13.1.	Normative References	94
13.2.	Informative References	98
Appendix A.	IPv4 and non-ANI operations	102
A.1.	IPv4 Link Local addresses	102
A.2.	Use of DHCPv4	102
Appendix B.	mDNS / DNSSD proxy discovery options	102
Appendix C.	Example Vouchers	103
C.1.	Keys involved	103
C.1.1.	Manufacturer Certificate Authority for IDevID signatures	104
C.1.2.	MASA key pair for voucher signatures	105
C.1.3.	Registrar Certificate Authority	107

C.1.4. Registrar key pair	108
C.1.5. Pledge key pair	110
C.2. Example process	111
C.2.1. Pledge to Registrar	111
C.2.2. Registrar to MASA	115
C.2.3. MASA to Registrar	121
Appendix D. Additional References	125
Authors' Addresses	125

1. Introduction

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices that are called pledges in this document. Pledges have an IDevID installed in them at the factory.

"BRSKI" is pronounced like "brewski", a colloquial term for beer in Canada and parts of the US-midwest. [brewski]

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]. This bootstrap process satisfies the [RFC7575] requirements of section 3.3 of making all operations secure by default. Other users of the BRSKI protocol will need to provide separate applicability statements that include privacy and security considerations appropriate to that deployment. Section 9 explains the detailed applicability for this the ACP usage.

The BRSKI protocol requires a significant amount of communication between manufacturer and owner: in its default modes it provides a cryptographic transfer of control to the initial owner. In its strongest modes, it leverages sales channel information to identify the owner in advance. Resale of devices is possible, provided that the manufacturer is willing to authorize the transfer. Mechanisms to enable transfers of ownership without manufacturer authorization are not included in this version of the protocol, but could be designed into future versions.

This document describes how pledges discover (or are discovered by) an element of the network domain to which the pledge belongs that will 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 this network?"

This document details protocols and messages to answer the above questions. It uses a TLS connection and an PKIX-shaped (X.509v3) certificate (an IEEE 802.1AR [IDevID] IDDevID) 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" (MASA) 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 several problems and limitations:

- * the pledge requires realtime connectivity to the manufacturer service,
- * the domain identity is exposed to the manufacturer service (this is a privacy concern),
- * 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined for clarity:

ANI: The Autonomic Network Infrastructure as defined by [I-D.ietf-anima-reference-model]. Section 9 details specific requirements for pledges, proxies and registrars when they are part of an ANI.

Circuit Proxy: A stateful implementation of the join proxy. This is the assumed type of proxy.

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.

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.

domainID: The domain IDentity is a unique value based upon the Registrar CA's certificate. Section 5.8.2 specifies how it is calculated.

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.

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.

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].

IDeVID: An Initial Device Identity X.509 certificate installed by the vendor on new equipment. This is a term from 802.1AR [IDeVID]

IPIP Proxy: A stateless proxy alternative.

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.

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".

LDeVID: A Local Device Identity X.509 certificate installed by the owner of the equipment. This is a term from 802.1AR [LDeVID]

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.

MASA Audit-Log: An anonymized list of previous owners maintained by

the MASA on a per device (per pledge) basis. Described in Section 5.8.1.

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.

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.

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.

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 accurate tracking of such ownership. Ownership tracking information is indicated in vouchers as described in [RFC8366]

Pledge: The prospective (unconfigured) device, which has an identity installed at the factory.

(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".

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.

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]

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+ [RFC7228]) 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 Low-power Lossy Networks (LLN)s).

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 intended benefit 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 are 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 Generic Autonomic Signaling Protocol (GRASP)'s DULL [I-D.ietf-anima-grasp] M_FLOOD announcements.

Upon successfully validating a voucher artifact, 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 Autonomic Service Agent (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.

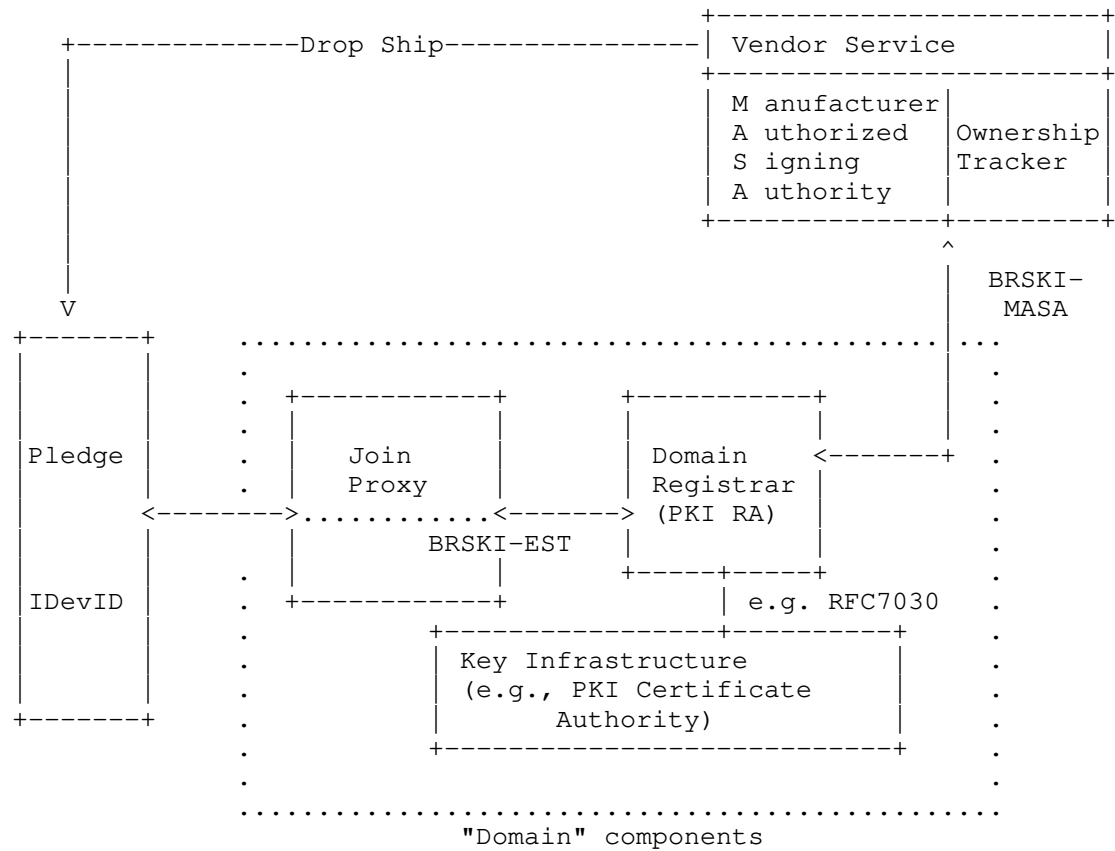


Figure 1: Architecture Overview

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 Certification Authority.

2.1. Behavior of a Pledge

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

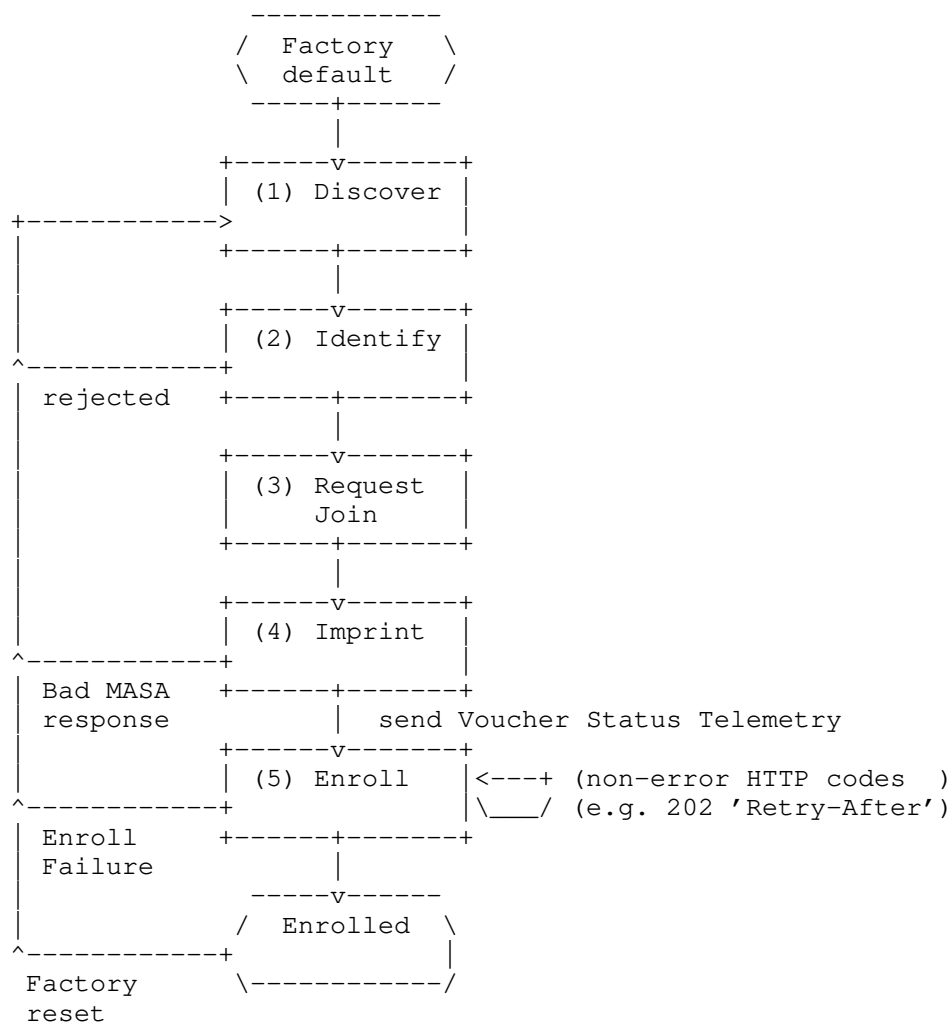


Figure 2: Pledge State Diagram

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] can then be 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 Representational State Transfer (REST) (see [REST]) interface could be integrated in future work.

2.2. Secure Imprinting using Vouchers

A voucher is a cryptographically protected artifact (using 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-shaped 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. Section 10.3 discusses privacy implications of the identifier.
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 (2009 edition) and section 8.10.3 (2018 edition) of [IDevID] discusses keyUsage and extendedKeyUsage extensions in the IDevID certificate. [IDevID] acknowledges that adding restrictions in the certificate limits applicability of these long-lived certificates. This specification emphasizes this point, and therefore RECOMMENDS that no key usage restrictions be included. This is consistent with [RFC5280] section 4.2.1.3, which does not require key usage restrictions for end entity certificates.

2.3.1. Identification of the Pledge

In the context of BRSKI, pledges have a 1:1 relationship with 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).

There is a (certificate) serialNumber field is defined in [RFC5280] section 4.1.2.2. In the ASN.1, this is referred to as the CertificateSerialNumber. This field is NOT relevant to this specification. Do not confuse this field with the "serial-number" defined by this document, or by [IDevID] and [RFC4519] section 2.31.

The device serial number is defined in [RFC5280] section A.1 and A.2 as the X520SerialNumber, with the OID tag id-at-serialNumber.

The device serial number field (X520SerialNumber) is 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".

An example of a printable form of the "serialNumber" field is provided in [RFC4519] section 2.31 ("WI-3005"). That section further provides equality and syntax attributes.

Due to the reality of existing device identity provisioning processes, some manufacturers have stored serial-numbers in other fields. Registrar's SHOULD be configurable, on a per-manufacturer basis, to look for serial-number equivalents in other fields.

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.

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].

The URI provides the authority information. The BRSKI `"/.well-known"` tree ([RFC5785]) is described in Section 5.

A complete URI MAY be in this extension, including the `'scheme'`, `'authority'`, and `'path'`. The complete URI will typically be used in diagnostic or experimental situations. Typically, (and in consideration to constrained systems), this SHOULD be reduced to only the `'authority'`, in which case a scheme of `"https://"` ([RFC7230] section 2.7.3) and `'path'` of `"/.well-known/brski"` is to be assumed.

The registrar can assume that only the `'authority'` 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:

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

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

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

id-pe FROM PKIX1Explicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51) } ;

MASACertExtensions EXTENSION ::= { ext-MASAURL, ... }
ext-MASAURL EXTENSION ::= { SYNTAX MASAURLSyntax
IDENTIFIED BY id-pe-masa-url }

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

MASAURLSyntax ::= IA5String

END
<CODE ENDS>
```

Figure 3: MASAURL ASN.1 Module

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 4

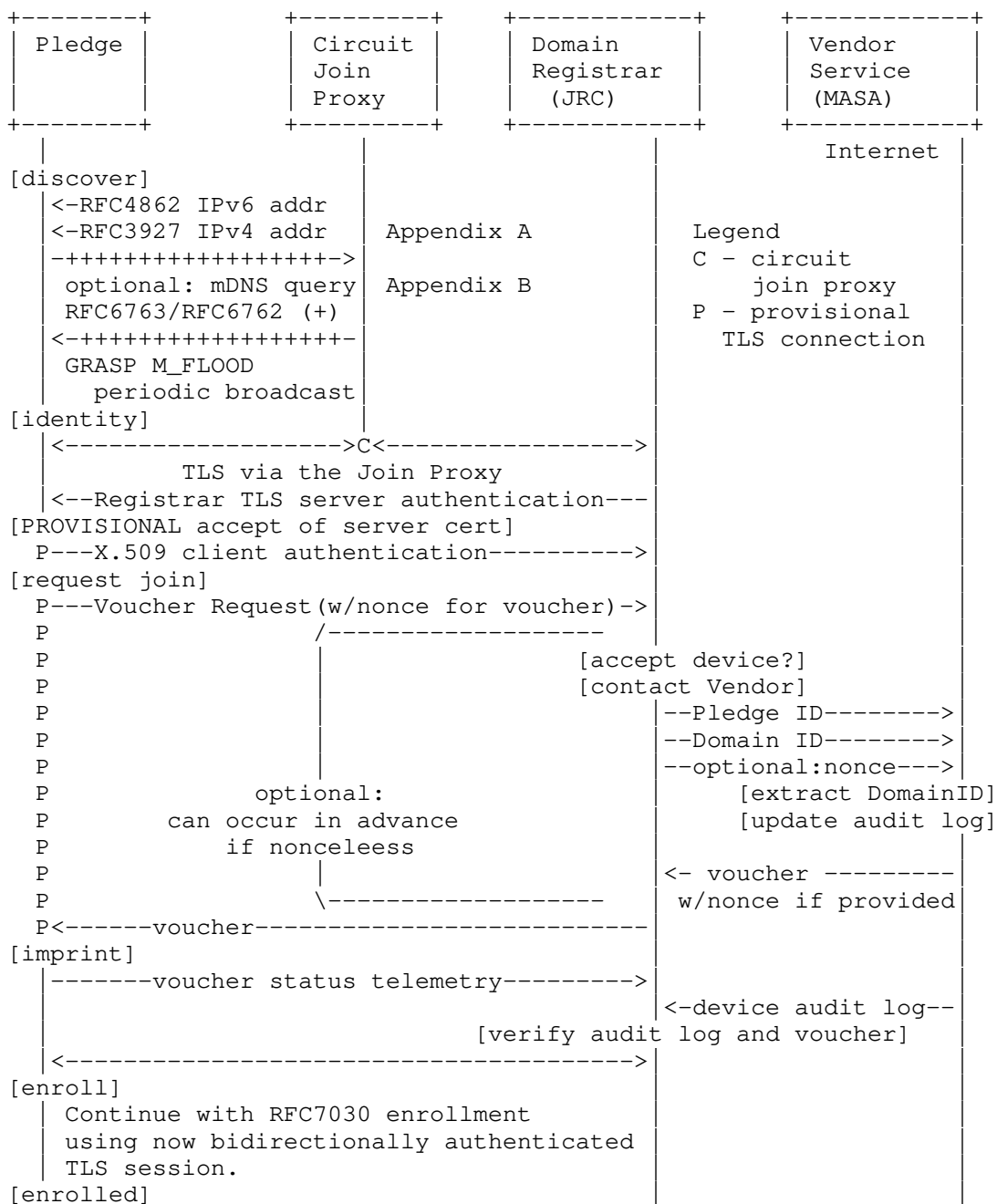


Figure 4: Protocol Time Sequence Diagram

On initial bootstrap, a new device (the pledge) uses a local service autodiscovery (GRASP or mDNS) to locate a join proxy. The join proxy connects the pledge to a local registrar (the JRC).

Having found a candidate registrar, the fledgling pledge sends some information about itself to the registrar, including its serial number in the form of a voucher request and its device identity certificate (IDevID) as part of the TLS session.

The registrar can determine whether it expected such a device to appear, and locates a MASA. The location of the MASA is usually found in an extension in the IDevID. Having determined that the MASA is suitable, the entire information from the initial voucher request (including device serial number) is transmitted over the internet in a TLS protected channel to the manufacturer, along with information about the registrar/owner.

The manufacturer can then apply policy based on the provided information, as well as other sources of information (such as sales records), to decide whether to approve the claim by the registrar to own the device; if the claim is accepted, a voucher is issued that directs the device to accept its new owner.

The voucher is returned to the registrar, but not immediately to the device -- the registrar has an opportunity to examine the voucher, the MASA's audit-logs, and other sources of information to determine whether the device has been tampered with, and whether the bootstrap should be accepted.

No filtering of information is possible in the signed voucher, so this is a binary yes-or-no decision. If the registrar accepts the voucher as a proper one for its device, the voucher is returned to the pledge for imprinting.

The voucher also includes a trust anchor that the pledge uses as representing the owner. This is used to successfully bootstrap from an environment where only the manufacturer has built-in trust by the device into an environment where the owner now has a PKI footprint on the device.

When BRSKI is followed with EST this single footprint is further leveraged into the full owner's PKI and a LDevID for the device. Subsequent reporting steps provide flows of information to indicate success/failure of the process.

2.5. Architectural Components

2.5.1. Pledge

The pledge is the device that is attempting to join. The pledge is assumed to talk to the Join Proxy using link-local network connectivity. In most cases, the pledge has no other connectivity until the pledge completes the enrollment process and receives some kind of network credential.

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 connection's MASA TLS Server Certificate. Configuration or distribution of trust anchors is out-of-scope for this specification.

The registrar uses a different Implicit Trust Anchor database for authenticating the BRSKI-EST connection's Pledge TLS Client Certificate. Configuration or distribution of the BRSKI-EST client trust anchors is out-of-scope of this specification. Note that the trust anchors in/excluded from the database will affect which manufacturers' devices are acceptable to the registrar as pledges, and can also be used to limit the set of MASAs that are trusted for enrollment.

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 concern, 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 with a framework 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.

A pledge with a real time clock in which it has confidence, MUST check the above time fields in all certificates and signatures that it processes.

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. Infinite Lifetime of IDevID

[RFC5280] explains that long lived pledge certificates "SHOULD be assigned the GeneralizedTime value of 99991231235959Z" for the notAfter field.

Some deployed IDevID management systems are not compliant with the 802.1AR requirement for infinite lifetimes, and put in typical ≤ 3 year certificate lifetimes. Registrars SHOULD be configurable on a per-manufacturer basis to ignore pledge lifetimes when the pledge did not follow the RFC5280 recommendations.

2.7. Cloud Registrar

There exist operationally open networks 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. The case where a device can boot and get access to larger Internet are less likely within the ANIMA ACP scope but may be more important in the future. In the ANIMA ACP scope, new devices will be quarantined behind a Join Proxy.

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 a manufacturer-assigned Implicit Trust Anchor database (see [RFC7030]) MUST be used to authenticate that service as described in [RFC6125]. The use of a DNS-ID for validation is appropriate, and it may include wildcard components on the left-mode side. 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.

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 pledges associated with each trust anchor.

3. Voucher-Request artifact

Voucher-requests are how vouchers are requested. The semantics of the voucher-request are described below, in the YANG model.

A pledge forms the "pledge voucher-request", signs it with its IDevID and submits it to the registrar.

The registrar in turn forms the "registrar voucher-request", signs it with its Registrar keypair 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.

This network proximity results from the following properties in the ACP context: the pledge is connected to the Join Proxy (Section 4) using a Link-Local IPv6 connection. While the Join Proxy does not participate in any meaningful sense in the cryptography of the TLS connection (such as via a Channel Binding), the Registrar can observe that the connection is via the private ACP (ULA) address of the join proxy, and could not come from outside the ACP. The Pledge must therefore be at most one IPv6 Link-Local hop away from an existing node on the ACP.

Other users of BRSKI will need to define other kinds of assertions if the network proximity described above does not match their needs.

The "prior-signed-voucher-request" leaf is used in registrar voucher-requests. If present, it is the signed pledge voucher-request artifact. 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 bar-code 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

```

Figure 5: YANG Tree diagram for Voucher-Request

3.3. Examples

This section provides voucher-request examples for illustration purposes. These examples show the JSON prior to CMS wrapping. JSON encoding rules specify that any binary content be base64 encoded ([RFC4648] section 4). The contents of the (base64) encoded certificates have been elided to save space. For detailed examples, see Appendix C.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": {
    "assertion": "proximity",
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",
    "serial-number" : "JADA123456789",
    "created-on": "2017-01-01T00:00:00.000Z",
    "proximity-registrar-cert": "base64encodedvalue=="
  }
}
```

Figure 6: JSON representation of example Voucher-Request

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 CMS signed object. In the JSON encoding used here it must be base64 encoded. The nonce and assertion have been carried forward from the pledge request to the registrar request. The serial-number is extracted from the pledge's Client Certificate from the TLS connection. See Section 5.5.

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

Figure 7: JSON representation of example Prior-Signed Voucher-Request

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"
  }
}
```

Figure 8: JSON representation of Offline Voucher-Request

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

  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 vch;
    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 8366: Voucher Artifact for
      Bootstrapping Protocols";
  }

  organization
```

"IETF ANIMA Working Group";

contact

"WG Web: <<https://datatracker.ietf.org/wg/anima/>>
WG List: <<mailto:anima@ietf.org>>
Author: Kent Watsen
<<mailto:kent+ietf@watsen.net>>
Author: Michael H. Behringer
<<mailto:Michael.H.Behringer@gmail.com>>
Author: Toerless Eckert
<<mailto:tte+ietf@cs.fau.de>>
Author: Max Pritikin
<<mailto:pritikin@cisco.com>>
Author: Michael Richardson
<<mailto:mcr+ietf@sandelman.ca>>";

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', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

Copyright (c) 2019 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: Bootstrapping Remote Secure Key Infrastructure";  
}
```

```
// 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 vch:voucher-artifact-grouping {
    refine "voucher/created-on" {
      mandatory false;
    }

    refine "voucher/pinned-domain-cert" {
      mandatory false;
      description "A pinned-domain-cert field
                   is not valid in a voucher request, and
                   any occurrence MUST be ignored";
    }

    refine "voucher/last-renewal-date" {
      description "A last-renewal-date field
                   is not valid in a voucher request, and
                   any occurrence MUST be ignored";
    }

    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 registrar 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 included in this field.

For example, a pledge might sign a voucher request with a proximity-registrar-cert, and the registrar then includes it as 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.X690.1994].

The first certificate in the Registrar TLS server certificate_list sequence (the end-entity TLS certificate, see [RFC8446]) presented by the Registrar to the Pledge.

This MUST be populated in a Pledge's voucher request when a proximity assertion is requested.";

}

}

}

}

}

<CODE ENDS>

Figure 9: YANG module for Voucher-Request

4. Proxying details (Pledge - Proxy - Registrar)

This section is normative for uses with an ANIMA ACP. The use of the GRASP mechanism is part of the ACP. Other users of BRSKI will need to define an equivalent proxy mechanism, and an equivalent mechanism to configure the proxy.

The role of the proxy is to facilitate communications. The proxy forwards packets between the pledge and a registrar that has been provisioned to the proxy via full GRASP ACP discovery.

This section defines a stateful proxy mechanism which is referred to as a "circuit" proxy. This is a form of Application Level Gateway ([RFC2663] section 2.9).

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. Please see [RFC8446] section 9.3 for details on TLS invariants.

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 proxying occurring. This is useful for Registrars which are 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 and Join Proxy 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 non-normative other methods (mDNS, and IPv4 methods) described in Appendix B are active. The pledge SHOULD run those methods in parallel with listening to for the M_FLOOD. The active methods SHOULD back-off by doubling 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 back off timers.

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 valid 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. Connection attempts to different proxies SHOULD be sent with an interval of 3 to 5s. The pledge SHOULD continue to listen to for additional GRASP M_FLOOD messages during the connection attempts.

Each connection attempt through a distinct Join Proxy MUST have a unique nonce in the voucher-request.

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 any 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 formal Concise Data Definition Language (CDDL) [RFC8610] definition is:

```

<CODE BEGINS> file "proxygrasp.cddl"
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
<CODE ENDS>

```

Figure 10: CDDL definition of Proxy Discovery message

Here is an example M_FLOOD announcing a proxy at fe80::1, on TCP port 4443.

```

[M_FLOOD, 12340815, h'fe800000000000000000000000000001', 180000,
  [{"AN_Proxy", 4, 1, ""},
   [O_IPv6_LOCATOR,
    h'fe800000000000000000000000000001', IPPROTO_TCP, 4443]]]

```

Figure 11: Example of Proxy Discovery message

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 its 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, with the "AN_join_registrar" objective. A registrar may announce any convenient port number, including using a stock port 443. ANI proxies MUST support GRASP discovery of registrars.

The M_FLOOD is formatted as follows:

```
[M_FLOOD, 51804321, h'fda379a6f6ee00000200000064000001', 180000,
  [{"AN_join_registrar", 4, 255, "EST-TLS"},
  [O_IPv6_LOCATOR,
   h'fda379a6f6ee00000200000064000001', IPPROTO_TCP, 8443]]]
```

Figure 12: An example of a Registrar announcement message

The formal CDDL definition is:

```
<CODE BEGINS> file "jrcgrasp.cddl"
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.
<CODE ENDS>
```

Figure 13: CDDL definition for Registrar announcement message

The M_FLOOD message MUST be sent periodically. The default period SHOULD be 60 seconds, the value SHOULD be operator configurable but SHOULD NOT be 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.

Here are some examples of locators for illustrative purposes. Only the first one (\$transport-protocol = 6, TCP) is defined in this document and is mandatory to implement.

```
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]
```

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 "/.well-known/brski" URI tree, as well as implementing 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 a new communication channel, similar to EST, within the newly registered `"/.well-known/brski"` tree. For clarity this channel is referred to as `"BRSKI-MASA"`. (See Figure 1).

The MASA URI is `"https://"` authority `"/.well-known/brski"`.

BRSKI uses existing CMS message formats for existing EST operations. BRSKI uses JSON [RFC8259] for all new operations defined here, and voucher formats. In all places where a binary value must be carried in a JSON string, the use of base64 format ([RFC4648] section 4) is to be used, as per [RFC7951] section 6.6.

While EST section 3.2 does not insist upon use of HTTP persistent connections ([RFC7230] section 6.3), 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.

If non-persistent connections are used, then both the pledge and the registrar MUST remember the certificates seen, and also sent for the first connection. They MUST check each subsequent connections for the same certificates, and each end MUST use the same certificates as well. This places a difficult restriction on rolling certificates on the Registrar.

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 a voucher using the new REST calls described below. This voucher is then validated.
- * 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 as explained in Section 2.3.1
- * The registrar requests and validates the voucher from the MASA.
- * The registrar forwards the voucher to the pledge when requested.
- * The registrar performs log verifications (described in Section 5.8.3) 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.

Use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED on the Pledge side. TLS 1.3 (or newer) SHOULD be available on the Registrar server interface, and the Registrar client interface, but TLS 1.2 MAY be used. TLS 1.3 (or newer) SHOULD be available on the MASA server interface, but TLS 1.2 MAY be used.

Establishment of the BRSKI-EST TLS connection is as specified in EST [RFC7030] section 4.1.1 "Bootstrap Distribution of CA Certificates" [RFC7030] wherein the client is authenticated with the IDevID certificate, and the EST server (the registrar) is provisionally authenticated with an unverified server certificate. Configuration or distribution of the trust anchor database used for validating the IDevID certificate is out-of-scope of this specification. Note that the trust anchors in/excluded from the database will affect which manufacturers' devices are acceptable to the registrar as pledges, and can also be used to limit the set of MASAs that are trusted for enrollment.

The signature in the certificate MUST be validated even if a signing key can not (yet) be validated. The certificate (or chain) MUST be retained for later validation.

A self-signed certificate for the Registrar is acceptable as the voucher can validate it upon successful enrollment.

The pledge performs input validation of all data received until a voucher is verified as specified in Section 5.6.1 and the TLS connection leaves the provisional state. Until these operations are complete the pledge could be communicating with an attacker.

The pledge code needs to be written with the assumption that all data is being transmitted at this point to an unauthenticated peer, and that received data, while inside a TLS connection, **MUST** be considered untrusted. This particularly applies to HTTP headers and CMS structures that make up the voucher.

A pledge that can connect to multiple Registrars 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 will need to rotate among the various choices, terminating connections that do not appear to be making progress. If no connection is making progress 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/brski/requestvoucher"`.

The pledge voucher-request Content-Type is:

`application/voucher-cms+json` [RFC8366] defines a "YANG-defined JSON document that has been signed using a CMS structure", and the voucher-request described in Section 3 is created in the same way. The media type is the same as defined in [RFC8366]. This is also used for the pledge voucher-request. The pledge **MUST** 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 field 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. Registrars and MASA 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 with the current date and time in yang:date-and-time format. This provides additional information to the MASA. Pledges that have no real-time clocks MAY omit this field.

nonce: The pledge voucher-request MUST contain a cryptographically strong random or pseudo-random number nonce (see [RFC4086] section 6.2). As the nonce is usually generated very early in the boot sequence there is a concern that the same nonce might be generated across multiple boots, or after a factory reset. Different nonces MUST be generated for each bootstrapping attempt, whether in series or concurrently. The freshness of this nonce mitigates against the lack of real-time clock as explained in Section 2.6.1.

assertion: The pledge indicates support for the mechanism described in this document, by putting the value "proximity" in the voucher-request, MUST include the "proximity-registrar-cert" field (below).

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. That is, it is the end-entity certificate. This MUST be populated in a pledge voucher-request.

serial-number The serial number of the pledge is included in the voucher-request from the Pledge. This value is included as a sanity check only, but it is not to be forwarded by the Registrar as described in Section 5.5.

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 confirms that the assertion is 'proximity' and that pinned 'proximity-registrar-cert' is the Registrar's certificate. If this validation fails, then there is an On-Path Attacker (MITM), and the connection MUST be closed after the returning an HTTP 401 error code.

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. For different networks, examples of automated acceptance may 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 validation fails the registrar SHOULD respond with the HTTP 404 error code. If the voucher-request is in an unknown format, then an HTTP 406 error code is more appropriate. A situation that could be resolved with administrative action (such as adding a vendor to a whitelist) MAY be responded with an 403 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. The mechanisms in [RFC6125] SHOULD be used in authentication of the MASA using a DNS-ID that matches that which is found in the IDevID. Registrars MAY include a mechanism to override the MASA URL on a manufacturer-by-manufacturer basis, and within that override it is appropriate to provide alternate anchors. This will typically be used by some vendors to establish explicit (or private) trust anchors for validating their MASA that is part of a sales channel integration.

Use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED. TLS 1.3 (or newer) SHOULD be available.

As described in [RFC7030], the MASA and the registrars SHOULD be prepared to support TLS client certificate authentication and/or HTTP Basic, Digest, or SCRAM authentication. This connection MAY also have no client authentication at all.

Registrars SHOULD permit trust anchors to be pre-configured on a per-vendor(MASA) basis. Registrars SHOULD include the ability to configure a TLS ClientCertificate on a per-MASA basis, or to use no client certificate. Registrars SHOULD also permit HTTP Basic and Digest authentication to be configured.

The authentication of the BRSKI-MASA connection does not change the voucher-request process, as voucher-requests are already signed by the registrar. Instead, this authentication provides access control to the audit-log as described in Section 5.8.

Implementors are advised that contacting the MASA is to establish a secured API connection with a web service and that there are a number of authentication models being explored within the industry. Registrars are RECOMMENDED to fail gracefully and generate useful administrative notifications or logs in the advent of unexpected HTTP 401 (Unauthorized) responses from the MASA.

5.4.1. MASA authentication of customer Registrar

Providing per-customer options requires that the customer's registrar be uniquely identified. This can be done by any stateless method that HTTPS supports such as with HTTP Basic or Digest authentication (that is using a password), but the use of TLS Client Certificate authentication is RECOMMENDED.

Stateful methods involving API tokens, or HTTP Cookies, are not recommended.

It is expected that the setup and configuration of per-customer Client Certificates is done as part of a sales ordering process.

The use of public PKI (i.e. WebPKI) End-Entity Certificates to identify the Registrar is reasonable, and if done universally this would permit a MASA to identify a customers' Registrar simply by a FQDN.

The use of DANE records in DNSSEC signed zones would also permit use of a FQDN to identify customer Registrars.

A third (and simplest, but least flexible) mechanism would be for the MASA to simply store the Registrar's certificate pinned in a database.

A MASA without any supply chain integration can simply accept Registrars without any authentication, or can accept them on a blind Trust-on-First-Use basis as described in Section 7.4.2.

This document does not make a specific recommendation on how the MASA authenticates the Registrar as there are likely different tradeoffs in different environments and product values. Even within the ANIMA ACP applicability, there is a significant difference between supply chain logistics for \$100 CPE devices and \$100,000 core routers.

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 interface ([RFC7231]).

This is done with an HTTP POST using the operation path value of `"/.well-known/brski/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.

MASA implementations SHOULD anticipate future media ntypes but of course will simply fail the request if those types are not yet known.

The voucher-request CMS object includes some number of certificates that are input to the MASA as it populates the `'pinned-domain-cert'`. As the [RFC8366] is quite flexible in what may be put into the `'pinned-domain-cert'`, the MASA needs some signal as to what certificate would be effective to populate the field with: it may range from the End Entity (EE) Certificate that the Registrar uses, to the entire private Enterprise CA certificate. More specific certificates result in a tighter binding of the voucher to the domain, while less specific certificates result in more flexibility in how the domain is represented by certificates.

A Registrar which is seeking a nonceless voucher for later offline use benefits from a less specific certificate, as it permits the actual keypair used by a future Registrar to be determined by the pinned certificate authority.

In some cases, a less specific certificate, such a public WebPKI certificate authority, could be too open, and could permit any entity issued a certificate by that authority to assume ownership of a device that has a voucher pinned. Future work may provide a solution to pin both a certificate and a name that would reduce such risk of malicious ownership assertions.

The Registrar SHOULD request a voucher with the most specificity consistent with the mode that it is operating in. In order to do this, when the Registrar prepares the CMS structure for the signed voucher-request, it SHOULD include only certificates which are part of the chain that it wishes the MASA to pin. This MAY be as small as only the End-Entity certificate (with id-kp-cmcRA set) that it uses as it's TLS Server Certificate, or it MAY be the entire chain, including the Domain CA.

The Registrar SHOULD include an [RFC7231] section 5.3.2 "Accept" header field 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. The MASA is expected to be flexible in what it accepts.

The registrar populates the voucher-request fields as follows:

created-on: The Registrars SHOULD populate this field with the current date and time when the Registrar formed this voucher request. This field provides additional information to the MASA.

nonce: This value, if present, is copied 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 Issuer value from the pledge IDevID certificate is included to ensure unique interpretation of the serial-number. In the case of nonceless (offline) voucher-request, then an appropriate value needs to be configured from the same out-of-band source as the serial-number.

prior-signed-voucher-request: The signed pledge voucher-request SHOULD be included in the registrar voucher-request. The entire CMS signed structure is to be included, base64 encoded for transport in the JSON structure.

A nonceless registrar voucher-request MAY be submitted to the MASA. Doing so allows the registrar to request a voucher when the pledge is offline, 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.

The "proximity-registrar-cert" field MUST NOT be present 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 MAY be unknown 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 (that is, a Registrar with the same Domain CA) 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 5.5.4 and Section 5.5.3). More flexible methods will likely involve sales channel integration and authorizations (details are out-of-scope of this document).

5.5.2. MASA pinning of registrar

A certificate chain is extracted from the Registrar's signed CMS container. This chain may be as short as a single End-Entity Certificate, up to the entire registrar certificate chain, including the Domain CA certificate, as specified in Section 5.5.

If the domain's CA is unknown to the MASA, then it is to be considered a temporary trust anchor for the rest of the steps in this section. The intention is not to authenticate the message as having come from a fully validated origin, but to establish the consistency of the domain PKI.

The MASA MAY use the certificate farthest in the chain chain that it received from the Registrar from the end-entity, as determined by MASA policy. A MASA MAY have a local policy that it only pins the End-Entity certificate. This is consistent with [RFC8366]. Details of the policy will typically depend upon the degree of Supply Chain Integration, and the mechanism used by the Registrar to authenticate. Such a policy would also determine how the MASA will respond to a request for a nonceless voucher.

5.5.3. MASA checking of voucher request signature

As described in Section 5.5.2, the MASA has extracted Registrar's domain CA. This is used to validate the CMS signature ([RFC5652]) on the voucher-request.

Normal PKIX revocation checking is assumed during voucher-request signature validation. This CA certificate MAY have Certificate Revocation List distribution points, or Online Certificate Status Protocol (OCSP) information ([RFC6960]). If they are present, the MASA MUST be able to reach the relevant servers belonging to the Registrar's domain CA to perform the revocation checks.

The use of OCSP Stapling is preferred.

5.5.4. MASA verification of domain registrar

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.

Even when a domain CA is authenticated to the MASA, and there is strong sales channel integration to understand who the legitimate owner is, the above id-kp-cmcRA check prevents arbitrary End-Entity certificates (such as an LDevID certificate) from having vouchers issued against them.

Other cases of inappropriate voucher issuance are detected by examination of the audit log.

If a nonceless voucher-request is submitted the MASA MUST authenticate the registrar as described in either EST [RFC7030] section 3.2.3, section 3.3.2, or by validating the registrar's certificate used to sign the registrar voucher-request using a configured trust anchor. 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's identity (via TLS Client Certificate or HTTP authentication) MAY be omitted if the device policy is to accept audit-only vouchers.

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 consistency check described above is checking that the 'proximity-registrar-cert' SPKI fingerprint exists within the registrar voucher-request CMS signature's certificate chain. This is substantially the same as the pin validation described in in [RFC7469] section 2.6, paragraph three.

If these checks succeed the MASA updates the voucher and audit-log assertion leafs with the "proximity" assertion, as defined by [RFC8366] section 5.3.

5.5.6. 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 exists, 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 5.5.4).

The MASA populates the audit-log with the nonce that was verified. If a nonceless voucher is issued, then the audit-log is to be populated with the JSON value "null".

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.

When a voucher request arrives at the registrar, if it has a cached response from the MASA for the corresponding registrar voucher-request, that cached response can be used according to local policy; otherwise the registrar constructs a new registrar voucher-request and sends it to the MASA.

Registrar evaluation of the voucher itself is purely for transparency and audit purposes to further inform log verification (see Section 5.8.3) 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 [RFC7230] error code when a problem occurs. In this case, the response data from the MASA MUST be a plaintext human-readable (UTF-8) 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 (blinking 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 field 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.

A pledge that retries a request after receiving a 202 message MUST resend the same voucher-request. It MUST NOT sign a new voucher-request each time, and in particular, it MUST NOT change the nonce value.

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 origin. 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: header fields, 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: value that is not understood.

The voucher response format is as indicated in the submitted Accept header fields 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]. Figure 14 shows a sample of the contents of a voucher.

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

Figure 14: An example voucher

The MASA populates the voucher fields as follows:

nonce: The nonce from the pledge if available. See Section 5.5.6.

assertion: The method used to verify the relationship between pledge and registrar. See Section 5.5.5.

pinned-domain-cert: A certificate. See Section 5.5.2. This figure is illustrative, for an example, see Appendix C.2 where an End Entity certificate is used.

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 sufficiently to generate the response as described in Section 5.8.1. The internal state requirements to maintain the audit-log are out-of-scope.

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 the nonce information in the voucher. If present, the nonce in the voucher must match the nonce the pledge submitted to the registrar; vouchers with no nonce can also be accepted (according to local policy, 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. Such a thing indicates a failure to enroll in this domain, and the pledge MUST attempt joining with other available Join Proxy.

The pledge MUST be prepared to ignore additional fields that it does not recognize.

5.6.2. Pledge authentication of provisional TLS connection

Following the process described in [RFC8366], the pledge should consider the public key from the pinned-domain-cert as the sole temporary trust anchor.

The pledge then evaluates the TLS Server Certificate chain that it received when the TLS connection was formed using this trust anchor. It is possible that the pinned-domain-cert matches the End-Entity Certificate provided in the TLS Server.

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 unsuccessful attempts with other proxies. Attempts should be made repeated at intervals according to the backoff timer described earlier. Attempts SHOULD be repeated as failure may be the result of a temporary inconsistency (an inconsistently rolled registrar key, or some other mis-configuration). The inconsistency 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 a 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 BRSKI 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.

The pledge MUST indicate its pledge status regarding the voucher. It does this by sending a status message to the Registrar.

The posted data media type: application/json

The client sends an HTTP POST to the server at the URI ".well-known/brski/voucher_status".

The format and semantics described below are for version 1. A version field is included to permit significant changes to this feedback in the future. A Registrar that receives a status message with a version larger than it knows about SHOULD log the contents and alert a human.

The Status field indicates if the voucher was acceptable. Boolean values are acceptable, where "true" indicates the voucher was acceptable.

If the voucher 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 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.

The version and status fields MUST be present. The Reason field SHOULD be present whenever the status field is false. The Reason-Context field is optional. In the case of a SUCCESS the Reason string MAY be omitted.

The keys to this JSON object are case-sensitive and MUST be lowercase. Figure 16 shows an example JSON.

```
<CODE BEGINS> file "voucherstatus.cddl"
voucherstatus-post = {
  "version": uint,
  "status": bool,
  ? "reason": text,
  ? "reason-context" : { $$arbitrary-map }
}
<CODE ENDS>
```

Figure 15: CDDL for voucher status POST

```
{
  "version": 1,
  "status": false,
  "reason": "Informative human readable message",
  "reason-context": { "additional" : "JSON" }
}
```

Figure 16: Example Status Telemetry

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.

Additional standard JSON fields in this POST MAY be added, see Section 8.5. A server that sees unknown fields should log them, but otherwise ignore them.

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 POST using the operation path value of `"/.well-known/brski/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.

It is reasonable for a Registrar, that the MASA does not believe to be the current owner, to request the audit-log. There are probably reasons for this which are hard to predict in advance. For instance, such a registrar may not be aware that the device has been resold; it may be that the device has been resold inappropriately, and this is how the original owner will learn of the occurrence. It is also possible that the device legitimately spends time in two different networks.

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") response ([RFC7231] sections 6.3.2 and 7.1), with the URL to the prepared (and idempotent, therefore cachable) audit response in the Location: header field.

In order to avoid enumeration of device audit-logs, MASA that return URLs SHOULD take care to make the returned URL unguessable. [W3C.WD-capability-urls-20140218] provides very good additional guidance. For instance, rather than returning URLs containing a database number such as <https://example.com/auditlog/1234> or the EUI of the device such as <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 device selected by the IDevID presented in the request. The audit log may be abridged by removal of old or repeated values as explained below. The returned data is in JSON format ([RFC8259]), and the Content-Type SHOULD be "application/json".

The following CDDL ([RFC8610]) explains the structure of the JSON format audit-log response:

```
<CODE BEGINS> file "auditlog.cddl"
audit-log-response = {
  "version": uint,
  "events": [ + event ]
  "truncation": {
    ? "nonced duplicates": uint,
    ? "nonceless duplicates": uint,
    ? "arbitrary": uint,
  }
}

event = {
  "date": text,
  "domainID": text,
  "nonce": text / null,
  "assertion": "verified" / "logged" / "proximity",
  ? "truncated": uint,
}
<CODE ENDS>
```

Figure 17: CDDL for audit-log response

An example:

```

{
  "version": "1",
  "events": [
    {
      "date": "2019-05-15T17:25:55.644-04:00",
      "domainID": "BduJhdHPpfhQLyponf48JzXSGZ8=",
      "nonce": "VOUFT-WwrEv0NuAQEHoV7Q",
      "assertion": "proximity",
      "truncated": "0"
    },
    {
      "date": "2017-05-15T17:25:55.644-04:00",
      "domainID": "BduJhdHPpfhQLyponf48JzXSGZ8=",
      "nonce": "f4G6Vilt8nKo/FieCVgpBg==",
      "assertion": "proximity"
    }
  ],
  "truncation": {
    "nonced duplicates": "0",
    "nonceless duplicates": "1",
    "arbitrary": "2"
  }
}

```

Figure 18: Example of audit-log response

The domainID is a binary SubjectKeyIdentifier value calculated according to Section 5.8.2. It is encoded once in base64 in order to be transported in this JSON container.

The date is in [RFC3339] format, which is consistent with typical JavaScript usage of JSON.

The truncation structure MAY be omitted if all values are zero. Any counter missing from the truncation structure is to be assumed to be zero.

The nonce is a string, as provided in the voucher-request, and used in the voucher. If no nonce was placed in the resulting voucher, then a value of null SHOULD be used in preference to omitting the entry. While the nonce is often created as a base64 encoded random series of bytes, this should not be assumed.

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 abridged 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 omitted. 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 abridged 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 omitted ("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.

A registrar that sees a version value greater than 1 indicates an audit log format that has been enhanced with additional information. No information will be removed in future versions; should an incompatible change be desired in the future, then a new HTTP end point will be used.

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 SHOULD anticipate new kinds of responses, and SHOULD provide operator controls to indicate how to process unknown responses.

5.8.2. Calculation of domainID

The domainID is a binary value (a BIT STRING) that uniquely identifies a Registrar by the "pinned-domain-cert".

If the "pinned-domain-cert" certificate includes the SubjectKeyIdentifier (Section 4.2.1.2 [RFC5280]), then it is to be used as the domainID. If not, the SPKI Fingerprint as described in [RFC7469] section 2.4 is to be used. This value needs to be calculated by both MASA (to populate the audit-log), and by the Registrar (to recognize itself in the audit log).

[RFC5280] section 4.2.1.2 does not mandate that the SubjectKeyIdentifier extension be present in non-CA certificates. It is RECOMMENDED that Registrar certificates (even if self-signed), always include the SubjectKeyIdentifier to be used as a domainID.

The domainID is determined from the certificate chain associated with the pinned-domain-cert and is used to update the audit-log.

5.8.3. Registrar audit log verification

Each time the Manufacturer Authorized Signing Authority (MASA) issues a voucher, it appends details of the assignment to an internal audit log for that device. The internal audit log is processed when responding to requests for details as 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: Nonceless 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. However, this entry does not indicate whether the pledge was actually deployed in the prior domain, or not. 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 require all vouchers to have a nonce. An alternative is to require that all nonces be from a subset (e.g. only internal) of domainIDs. If the policy is violated a simple action is to revoke any locally issued credentials for the pledge in question or to refuse to forward the voucher. The Registrar MUST then refuse any EST actions, and SHOULD inform a human via a log. A registrar MAY be configured to ignore (i.e. override the above policy) the history of the device but it is RECOMMENDED that this only be configured if hardware assisted (i.e. TPM anchored) Network Endpoint Assessment (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 API calls provide an automated alternative to the manual bootstrapping method described in [RFC7030]. As noted above, use of HTTP persistent connections simplifies the pledge state machine.

Although EST allows clients to obtain multiple certificates by sending multiple Certificate Signing Requests (CSR) requests, BRSKI does not support this mechanism directly. This is because BRSKI pledges MUST use the CSR Attributes request ([RFC7030] section 4.5). The registrar MUST 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 5.6.2 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 (such as rfc822Name). This approach is beneficial to automated bootstrapping in the widest number of environments.

In networks using the BRSKI enrolled certificate to authenticate the ACP (Autonomic Control Plane), the EST CSR attributes MUST include the ACP Domain Information Fields defined in [I-D.ietf-anima-autonomic-control-plane] section 6.1.1.

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. The 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.

The client MUST send an indicator to the Registrar about its enrollment status. It does this by using an HTTP POST of a JSON dictionary with the of attributes described below to the new EST endpoint at `"/.well-known/brski/enrollstatus"`. (XXX ?)

When indicating a successful enrollment the client SHOULD first re-establish the EST TLS session using the newly obtained credentials. TLS 1.2 supports doing this in-band, but TLS 1.3 does not. The client SHOULD therefore always close the existing TLS connection, and start a new one.

In the case of a failed enrollment, the client MUST send the telemetry information over the same TLS connection that was used for the enrollment attempt, with a Reason string indicating why the most recent enrollment failed. (For failed attempts, the TLS connection is the most reliable way to correlate server-side information with what the client provides.)

The version and status fields MUST be present. The Reason field SHOULD be present whenever the status field is false. In the case of a SUCCESS the Reason string MAY be omitted.

The reason-context attribute is an arbitrary JSON object (literal value or hash of values) which provides additional information specific to the failure to unroll from this pledge. The contents of this field are not subject to standardization. This is represented by the group-socket "\$\$arbitrary-map" in the CDDL.

In the case of a SUCCESS the Reason string is omitted.

```
<CODE BEGINS> file "enrollstatus.cddl"
enrollstatus-post = {
    "version": uint,
    "status": bool,
    ? "reason": text,
    ? "reason-context" : { $$arbitrary-map }
}
<CODE ENDS>
```

Figure 19: CDDL for enrollment status POST

An example status report can be seen below. It is sent with with the media type: application/json

```
{
    "version": 1,
    "status":true,
    "reason":"Informative human readable message",
    "reason-context": { "additional" : "JSON" }
}
```

Figure 20: Example of enrollment status POST

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.

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 end points which send and receive binary artifacts (specifically, `"/.well-known/brski/requestvoucher"`) are binary. That is, no encoding is used.

In the BRSKI context, the EST "Content-Transfer-Encoding" header field if present, SHOULD be ignored. This header field 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. This section suggests a range of mechanisms that would alter the security assurance of BRSKI to accommodate alternative deployment architectures and mitigate lifecycle management issues identified in Section 10. They are presented here as informative (non-normative) design guidance for future standardization activities. Section 9 provides standardization applicability statements for the ANIMA ACP. Other users would be expected that subsets of these mechanisms could be profiled with an accompanying applicability statements similar to the one described in Section 9.

This section is considered non-normative in the generality of the protocol. Use of the suggested mechanisms here MUST be detailed in specific profiles of BRSKI, such as in Section 9.

7.1. Trust Model

This section explains the trust relationships detailed in Section 2.4:

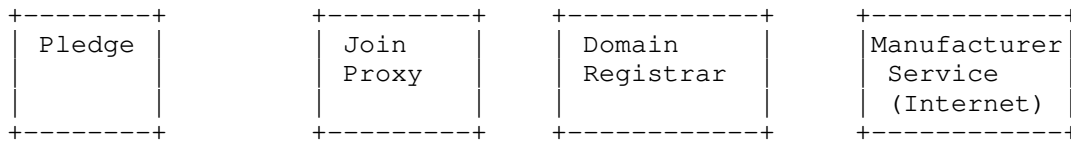


Figure 10

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.

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 following is a list of alternative behaviours that the pledge can be programmed to implement. These behaviours are not mutually exclusive, nor are they dependent upon each other. Some of these methods enable offline and emergency (touch based) deployment use cases. Normative language is used as these behaviours are referenced in later sections in a normative fashion.

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.
4. A craft/serial console could include a command such as "est-enroll [2001:db8:0:1]:443" that begins the EST process from the point after the voucher is validated. This process SHOULD include server certificate verification using an on-screen fingerprint.

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 (NEA: [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. They MUST NOT be the default behavior. These methods may be acceptable in situations where threat models indicate that low security is adequate. This includes situations where security decisions are being made by the local administrator:

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 offline 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 the lower-security behavior is tied to specific device identities. The modes described below can be applied to specific devices via knowledge of what devices were sold. They can also be bound to specific customers (independent of the device identity) by authenticating the customer's Registrar.

7.4.1. Issuing Nonceless vouchers

A MASA has the option of not including a nonce in the voucher, and/or not requiring one to be present in the voucher-request. This results in distribution of a voucher that may never expire and in effect makes the specified Domain an always trusted entity to the pledge during any subsequent bootstrapping attempts. That a nonceless voucher was issued is captured in the log information so that the

registrar can make appropriate security decisions when a pledge joins the Domain. Nonceless vouchers are useful to support use cases where registrars might not be online during actual device deployment.

While a nonceless voucher may include an expiry date, a typical use for a nonceless voucher is for it to be long-lived. If the device can be trusted to have an accurate clock (the MASA will know), then a nonceless voucher CAN be issued with a limited lifetime.

A more typical case for a nonceless voucher is for use with offline onboarding scenarios where it is not possible to pass a fresh voucher-request to the MASA. The use of a long-lived voucher also eliminates concern about the availability of the MASA many years in the future. Thus many nonceless vouchers will have no expiry dates.

Thus, the long lived nonceless voucher does not require the proof that the device is online. Issuing such a thing is only accepted when the registrar is authenticated by the MASA and the MASA is authorized to provide this functionality to this customer. The MASA is RECOMMENDED to use this functionality only in concert with an enhanced level of ownership tracking, the details of which are out of scope for this document.

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.

7.4.2. Trusting Owners on First Use

A MASA has the option of 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. The proof-of-proximity comes from the assumption that the pledge and Join Proxy are on the same link-local connection.

A MASA that practices Trust-on-First-Use (TOFU) for Registrar identity may wish to annotate the origin of the connection by IP address or netblock, and restrict future use of that identity from other locations. A MASA that does this SHOULD take care to not create nuisance situations for itself when a customer has multiple registrars, or uses outgoing IPv4 NAT44 connections that change frequently.

7.4.3. Updating or extending voucher trust anchors

This section deals with the problem of a MASA that is no longer available due to a failed business, or the situation where a MASA is uncooperative to a secondary sale.

A manufacturer could offer a management mechanism that allows the list of voucher verification trust anchors to be extended. [I-D.ietf-netconf-keystore] is one such interface that could be implemented using YANG. Pretty much any configuration mechanism used today could be extended to provide the needed additional update. A manufacturer could even decide to install the domain CA trust anchors received during the EST "cacerts" step as voucher verification anchors. Some additional signals will be needed to clearly identify which keys have voucher validation authority from among those signed by the domain CA. This is future work.

With the above change to the list of anchors, vouchers can be issued by an alternate MASA. This could be the previous owner (the seller), or some other trusted third party who is mediating the sale. If it was a third party, then the seller would need to have taken steps to introduce the third party configuration to the device prior disconnection. The third party (e.g. a wholesaler of used equipment) could however use a mechanism described in Section 7.2 to take control of the device after receiving it physically. This would permit the third party to act as the MASA for future onboarding actions. As the IDevID certificate probably can not be replaced, the new owner's Registrar would have to support an override of the MASA URL.

To be useful for resale or other transfers of ownership one of two situations will need to occur. The simplest is that the device is not put through any kind of factory default/reset before going through onboarding again. Some other secure, physical signal would be needed to initiate it. This is most suitable for redeploying a device within the same Enterprise. This would entail having previous configuration in the system until entirely replaced by the new owner, and represents some level of risk.

The second mechanism is that there would need to be two levels of factory reset. One would take the system back entirely to manufacturer state, including removing any added trust anchors, and the second (more commonly used) one would just restore the configuration back to a known default without erasing trust anchors. This weaker factory reset might leave valuable credentials on the device and this may be unacceptable to some owners.

As a third option, the manufacturer's trust anchors could be entirely overwritten with local trust anchors. A factory default would never restore those anchors. This option comes with a lot of power, but also a lot of responsibility: if access to the private part of the new anchors are lost the manufacturer may be unable to help.

8. IANA Considerations

This document requires the following IANA actions:

8.1. The IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. IANA is asked to register the following:

URI: urn:ietf:params:xml:ns:yang:ietf-voucher-request
Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

8.2. YANG Module Names Registry

This document registers a YANG module in the "YANG Module Names" registry [RFC6020]. IANA is asked to register the following:

name: ietf-voucher-request
namespace: urn:ietf:params:xml:ns:yang:ietf-voucher-request
prefix: vch
reference: THIS DOCUMENT

8.3. BRSKI well-known considerations

8.3.1. BRSKI .well-known registration

To the Well-Known URIs Registry, at:
"https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml", this document registers the well-known name "brski" with the following filled-in template from [RFC5785]:

URI suffix: brski
Change Controller: IETF

IANA is asked to change the registration of "est" to now only include RFC7030 and no longer this document. Earlier versions of this document used "/.well-known/est" rather than "/.well-known/brski".

8.3.2. BRSKI .well-known registry

IANA is requested to create a new Registry entitled: "BRSKI well-known URIs". The registry shall have at least three columns: URI, description, and reference. New items can be added using the Specification Required process. The initial contents of this registry shall be:

	URI	document	description
to MASA	requestvoucher	[THISRFC]	pledge to registrar, and from registrar
	voucher_status	[THISRFC]	pledge to registrar
	requestauditlog	[THISRFC]	registrar to MASA
	enrollstatus	[THISRFC]	pledge to registrar

8.4. 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.

8.5. 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 process. The following items are to be in the initial registration, with this document (Section 5.7) as the reference:

- * version
- * Status
- * Reason
- * reason-context

8.6. 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]

8.7. GRASP Objective Names

IANA is requested to register the following GRASP Objective Names:

The IANA is requested to register the value "AN_Proxy" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 4.1.1.

The IANA is requested to register the value "AN_join_registrar" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 4.3.

9. Applicability to the Autonomic Control Plane (ACP)

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 (ACP) that is bootstrapped by the BRSKI protocol is typically used in medium to large Internet Service Provider organizations. Equivalent enterprises that have significant layer-3 router connectivity also will find significant benefit, particularly if the Enterprise has many sites. (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).

In the ACP, the Join Proxy is found to be proximal because communication between the pledge and the join proxy is exclusively on IPv6 Link-Local addresses. The proximity of the Join Proxy to the Registrar is validated by the Registrar using ANI ACP IPv6 Unique Local Addresses (ULA). ULAs are not routable over the Internet, so as long as the Join Proxy is operating correctly the proximity assertion is satisfied. Other uses of BRSKI will need make similar analysis if they use proximity assertions.

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 is also capable of performing bootstrapping of a device using a serial-console (craft console). The zero-touch mechanism presented in this and the ACP document [I-D.ietf-anima-autonomic-control-plane] 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 may still have to visit. In recognition of this, some mechanisms are presented in Section 7.2. The manufacturer MUST provide at least one of the one-touch mechanisms described that permit enrollment to be proceed without availability of any manufacturer server (such as the MASA).

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.

9.1. Operational Requirements

This section collects operational requirements based upon the three roles involved in BRSKI: The Manufacturer Authorized Signing Authority (MASA), the (Domain) Owner and the Device. It should be recognized that the manufacturer may be involved in two roles, as it creates the software/firmware for the device, and also may be the operator of the MASA.

The requirements in this section are presented using BCP14 ([RFC2119], [RFC8174]) language. These do not represent new normative statements, just a review of a few such things in one place by role. They also apply specifically to the ANIMA ACP use case. Other use cases likely have similar, but MAY have different requirements.

9.1.1. MASA Operational Requirements

The manufacturer **MUST** arrange for an online service to be available called the MASA. It **MUST** be available at the URL which is encoded in the IDevID certificate extensions described in Section 2.3.2.

The online service **MUST** have access to a private key with which to sign [RFC8366] format voucher artifacts. The public key, certificate, or certificate chain **MUST** be built in to the device as part of the firmware.

It is **RECOMMENDED** that the manufacturer arrange for this signing key (or keys) to be escrowed according to typical software source code escrow practices [softwareescrow].

The MASA accepts voucher requests from Domain Owners according to an operational practice appropriate for the device. This can range from any domain owner (first-come first-served, on a TOFU-like basis), to full sales channel integration where Domain Owners need to be positively identified by TLS Client Certificate pinned, or HTTP Authentication process. The MASA creates signed voucher artifacts according to its internally defined policies.

The MASA **MUST** operate an audit log for devices that is accessible. The audit log is designed to be easily cacheable and the MASA **MAY** find it useful to put this content on a CDN.

9.1.2. Domain Owner Operational Requirements

The domain owner MUST operate an EST ([RFC7030]) server with the extensions described in this document. This is the JRC or Registrar. This JRC/EST server MUST announce itself using GRASP within the ACP. This EST server will typically reside with the Network Operations Center for the organization.

The domain owner MAY operate an internal certificate authority (CA) that is separate from the EST server, or it MAY combine all activities into a single device. The determination of the architecture depends upon the scale and resiliency requirements of the organization. Multiple JRC instances MAY be announced into the ACP from multiple locations to achieve an appropriate level of redundancy.

In order to recognize which devices and which manufacturers are welcome on the domain owner's network, the domain owner SHOULD maintain a white list of manufacturers. This MAY extend to integration with purchasing departments to know the serial numbers of devices.

The domain owner SHOULD use the resulting overlay ACP network to manage devices, replacing legacy out-of-band mechanisms.

The domain owner SHOULD operate one or more EST servers which can be used to renew the domain certificates (LDevIDs) which are deployed to devices. These servers MAY be the same as the JRC, or MAY be a distinct set of devices, as appropriate for resiliency.

The organization MUST take appropriate precautions against loss of access to the certificate authority private key. Hardware security modules and/or secret splitting are appropriate.

9.1.3. Device Operational Requirements

Devices MUST come with built-in trust anchors that permit the device to validate vouchers from the MASA.

Device MUST come with (unique, per-device) IDevID certificates that include their serial numbers, and the MASA URL extension.

Devices are expected to find Join Proxies using GRASP, and then connect to the JRC using the protocol described in this document.

Once a domain owner has been validated with the voucher, devices are expected to enroll into the domain using EST. Devices are then expected to form ACPs using IPsec over IPv6 Link-Local addresses as described in [I-D.ietf-anima-autonomic-control-plane].

Once a device has been enrolled it SHOULD listen for the address of the JRC using GRASP, and it SHOULD enable itself as a Join Proxy, and announce itself on all links/interfaces using GRASP DULL.

Devices are expected to renew their certificates before they expire.

10. Privacy Considerations

10.1. MASA audit log

The MASA audit log includes the domainID for each domain a voucher has been issued to. This information is closely related to the actual domain identity. A MASA may need additional defenses against Denial of Service attacks (Section 11.1), and this may involve collecting additional (unspecified here) information. 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-EST reveals

During the provisional phase of the BRSKI-EST connection between the Pledge and the Registrar, each party reveals its certificates to each other. For the Pledge, this includes the serialNumber attribute, the MASA URL, and the identity that signed the IDevID certificate.

TLS 1.2 reveals the certificate identities to on-path observers, including the Join Proxy.

TLS 1.3 reveals the certificate identities only to the end parties, but as the connection is provisional, an on-path attacker (MITM) can see the certificates. This includes not just malicious attackers, but also Registrars that are visible to the Pledge, but which are not part of the intended domain.

The certificate of the Registrar is rather arbitrary from the point of view of the BRSKI protocol. As no [RFC6125] validations are expected to be done, the contents could be easily pseudonymized. Any device that can see a join proxy would be able to connect to the Registrar and learn the identity of the network in question. Even if the contents of the certificate are pseudonymized, it would be possible to correlate different connections in different locations belong to the same entity. This is unlikely to present a significant privacy concern to ANIMA ACP uses of BRSKI, but may be a concern to other users of BRSKI.

The certificate of the Pledge could be revealed by a malicious Join Proxy that performed a MITM attack on the provisional TLS connection. Such an attacker would be able to reveal the identity of the Pledge to third parties if it chose to so.

Research into a mechanism to do multi-step, multi-party authenticated key agreement, incorporating some kind of zero-knowledge proof would be valuable. Such a mechanism would ideally avoid disclosing identities until pledge, registrar and MASA agree to the transaction. Such a mechanism would need to discover the location of the MASA without knowing the identity of the pledge, or the identity of the MASA. This part of the problem may be unsolveable.

10.3. What BRSKI-MASA reveals to the manufacturer

With consumer-oriented devices, the "call-home" mechanism in IoT devices raises significant privacy concerns. See [livingwithIoT] and [IoTstrangeThings] for exemplars. 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 where the "call-home" represents a different class of privacy and lifecycle management concerns.

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, but remarkably unusual, that a device could be forced to go through a full factory reset during an exceptional firmware update situation, after which enrollment would have to be repeated, and a new connection would occur)

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 is a mechanism to defend against exfiltration of data by a nefarious pledge. 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. This is revealed by transmission of a signed voucher-request containing the serial-number. The manufacturer can usually link the serial number to a device model.
- * 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 5.5.4.
- * 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.

Based upon the above information, the manufacturer is able to track a specific device from pseudonymous domain identity to the next pseudonymous domain identity. If there is sales-channel integration, then the identities are not pseudonymous.

The manufacturer knows the IP address of the Registrar, but it can not see the IP address of the device itself. The manufacturer can not track the device to a detailed physical or network location, only to the location of the Registrar. That is likely to be at the Enterprise or ISPs headquarters.

The above situation is to be distinguished from a residential/individual person who registers a device from a manufacturer. Individuals do not tend to have multiple offices, and their registrar is likely on the same network as the device. A manufacturer that sells switching/routing products to enterprises should hardly be surprised if additional purchases switching/routing products are made. Deviations from a historical trend or an establish baseline 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 re-designs, outsourcing, etc.) separate from the MASA, which may need to operate reliably for decades.

10.4. 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. In the absence of such manufacturer protection, 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.

10.5. 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.

10.6. 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.

Section 7.4.3 describes some ways in which a manufacturer could provide a mechanism to manage the trust anchors and built-in certificates (IDevID) as an extension. There are a variety of mechanism, and some may take a substantial amount of work to get exactly correct. These mechanisms do not change the flow of the protocol described here, but rather allow the starting trust assumptions to be changed. This is 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.

The BRSKI protocol depends upon a trust anchor on the device and an identity on the device. Management of these entities facilitates 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.

Although replacement of the IDevID is not required for all modes described above, a manufacturers could support such a thing. Some may wish to consider replacement of the IDevID as an indication that the device's warrantee is terminated. For others, the privacy requirements 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.

10.7. Death of a manufacturer

A common concern has been that a manufacturer could go out of business, leaving owners of devices unable to get new vouchers for existing products. Said products might have been previously deployed, but need to be re-initialized, they might have been purchased used, or they might have kept in a warehouse as long-term spares.

The MASA was named the Manufacturer *Authorized* Signing Authority to emphasize that it need not be the manufacturer itself that performs this. It is anticipated that specialist service providers will come

to exist that deal with the creation of vouchers in much the same way that many companies have outsourced email, advertising and janitorial services.

Further, it is expected that as part of any service agreement that the manufacturer would arrange to escrow appropriate private keys such that a MASA service could be provided by a third party. This has routinely been done for source code for decades.

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 Registrar 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. Denial of Service (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. Supply chain integration ("know your customer") is an additional step that MASA providers and device vendors can explore.

11.2. DomainID must be resistant to second-preimage attacks

The domainID is used as the reference in the audit log to the domain. The domainID is expected to be calculated by a hash that is resistant to a second-preimage attack. Such an attack would allow a second registrar to create audit log entries that are fake.

11.3. Availability of good random numbers

The nonce used by the Pledge in the voucher-request SHOULD be generated by a Strong Cryptographic Sequence ([RFC4086] section 6.2). TLS has a similar requirement.

In particular implementations should pay attention to the advance in [RFC4086] section 3, particularly section 3.4. The random seed used by a device at boot MUST be unique across all devices and all bootstraps. Resetting a device to factory default state does not obviate this requirement.

11.4. 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.

11.5. 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 mapping 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.

11.6. Manufacturer Maintenance 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 issues 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.

The set of standard operating procedures for maintaining high value private keys is well documented. For instance, the WebPKI provides a number of options for audits at [cabforumaudit], and the DNSSEC root operations are well documented at [dnssecroot].

It is not clear if Manufacturers will take this level of precaution, or how strong the economic incentives are to maintain an appropriate level of security.

This next section examines the risk due to a compromised manufacturer IDevID signing key. This is followed by examination of the risk due to a compromised MASA key. The third section sections below examines the situation where MASA web server itself is under attacker control, but that the MASA signing key itself is safe in a not-directly connected hardware module.

11.6.1. Compromise of Manufacturer IDevID signing keys

An attacker that has access to the key that the manufacturer uses to sign IDevID certificates can create counterfeit devices. Such devices can claim to be from a particular manufacturer, but be entirely different devices: Trojan horses in effect.

As the attacker controls the MASA URL in the certificate, the registrar can be convinced to talk to the attackers' MASA. The Registrar does not need to be in any kind of promiscuous mode to be vulnerable.

In addition to creating fake devices, the attacker may also be able to issue revocations for existing certificates if the IDevID certificate process relies upon CRL lists that are distributed.

There does not otherwise seem to be any risk from this compromise to devices which are already deployed, or which are sitting locally in boxes waiting for deployment (local spares). The issue is that operators will be unable to trust devices which have been in an uncontrolled warehouse as they do not know if those are real devices.

11.6.2. Compromise of MASA signing keys

There are two periods of time in which to consider: when the MASA key has fallen into the hands of an attacker, and after the MASA recognizes that the key has been compromised.

11.6.2.1. Attacker opportunities with compromised MASA key

An attacker that has access to the MASA signing key could create vouchers. These vouchers could be for existing deployed devices, or for devices which are still in a warehouse. In order to exploit these vouchers two things need to occur: the device has to go through a factory default boot cycle, and the registrar has to be convinced to contact the attacker's MASA.

If the attacker controls a Registrar which is visible to the device, then there is no difficulty in delivery of the false voucher. A possible practical example of an attack like this would be in a data center, at an ISP peering point (whether a public IX, or a private peering point). In such a situation, there are already cables attached to the equipment that lead to other devices (the peers at the IX), and through those links, the false voucher could be delivered. The difficult part would be get the device put through a factory reset. This might be accomplished through social engineering of data center staff. Most locked cages have ventilation holes, and possibly a long "paperclip" could reach through to depress a factory

reset button. Once such a piece of ISP equipment has been compromised, it could be used to compromise equipment that was connected to (through long haul links even), assuming that those pieces of equipment could also be forced through a factory reset.

The above scenario seems rather unlikely as it requires some element of physical access; but were there a remote exploit that did not cause a direct breach, but rather a fault that resulted in a factory reset, this could provide a reasonable path.

The above deals with ANI uses of BRSKI. For cases where 802.11 or 802.15.4 is involved, the need to connect directly to the device is eliminated, but the need to do a factory reset is not. Physical possession of the device is not required as above, provided that there is some way to force a factory reset. With some consumer devices with low overall implementation quality, the end users might be familiar with needing to reset the device regularly.

The authors are unable to come up with an attack scenario where a compromised voucher signature enables an attacker to introduce a compromised pledge into an existing operator's network. This is the case because the operator controls the communication between Registrar and MASA, and there is no opportunity to introduce the fake voucher through that conduit.

11.6.2.2. Risks after key compromise is known

Once the operator of the MASA realizes that the voucher signing key has been compromised it has to do a few things.

First, it MUST issue a firmware update to all devices that had that key as a trust anchor, such that they will no longer trust vouchers from that key. This will affect devices in the field which are operating, but those devices, being in operation, are not performing onboarding operations, so this is not a critical patch.

Devices in boxes (in warehouses) are vulnerable, and remain vulnerable until patched. An operator would be prudent to unbox the devices, onboard them in a safe environment, and then perform firmware updates. This does not have to be done by the end-operator; it could be done by a distributor that stores the spares. A recommended practice for high value devices (which typically have a <4hr service window) may be to validate the device operation on a regular basis anyway.

If the onboarding process includes attestations about firmware versions, then through that process the operator would be advised to upgrade the firmware before going into production. Unfortunately, this does not help against situations where the attacker operates their own Registrar (as listed above).

[RFC8366] section 6.1 explains the need for short-lived vouchers. The nonce guarantees freshness, and the short-lived nature of the voucher means that the window to deliver a fake voucher is very short. A nonceless, long-lived voucher would be the only option for the attacker, and devices in the warehouse would be vulnerable to such a thing.

A key operational recommendation is for manufacturers to sign nonceless, long-lived vouchers with a different key than they sign short-lived vouchers. That key needs significantly better protection. If both keys come from a common trust-anchor (the manufacturer's CA), then a compromise of the manufacturer's CA would compromise both keys. Such a compromise of the manufacturer's CA likely compromises all keys outlined in this section.

11.6.3. Compromise of MASA web service

An attacker that takes over the MASA web service has a number of attacks. The most obvious one is simply to take the database listing customers and devices and to sell this data to other attackers who will now know where to find potentially vulnerable devices.

The second most obvious thing that the attacker can do is to kill the service, or make it operate unreliably, making customers frustrated. This could have a serious affect on ability to deploy new services by customers, and would be a significant issue during disaster recovery.

While the compromise of the MASA web service may lead to the compromise of the MASA voucher signing key, if the signing occurs offboard (such as in a hardware signing module, HSM), then the key may well be safe, but control over it resides with the attacker.

Such an attacker can issue vouchers for any device presently in service. Said device still needs to be convinced to do through a factory reset process before an attack.

If the attacker has access to a key that is trusted for long-lived nonceless vouchers, then they could issue vouchers for devices which are not yet in service. This attack may be very hard to verify and as it would involve doing firmware updates on every device in warehouses (a potentially ruinously expensive process), a manufacturer might be reluctant to admit this possibility.

11.7. YANG Module Security Considerations

As described in the Security Considerations section of [RFC8366] (section 7.4), the YANG module specified in this document defines the schema for data that is subsequently encapsulated by a CMS signed-data content type, as described in Section 5 of [RFC5652]. As such, all of the YANG modeled data is protected from modification.

The use of YANG to define data structures, via the 'yang-data' statement, is relatively new and distinct from the traditional use of YANG to define an API accessed by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. For this reason, these guidelines do not follow template described by Section 3.7 of [RFC8407].

12. Acknowledgements

We would like to thank the various reviewers for their input, in particular William Atwood, Brian Carpenter, Fuyu Eleven, Eliot Lear, Sergey Kasatkin, Anoop Kumar, Tom Petch, Markus Stenberg, Peter van der Stok, and Thomas Werner

Significant reviews were done by Jari Arko, Christian Huitema and Russ Housley.

Henk Birkholz contributed the CDDL for the audit log response.

This document started it's life as a two-page idea from Steinthor Bjarnason.

In addition, significant review comments were received by many IESG members, including Adam Roach, Alexey Melnikov, Alissa Cooper, Benjamin Kaduk, Eric Vyncke, Roman Danyliw, and Magnus Westerlund.

13. References

13.1. Normative References

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

Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-30, 30 October 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-autonomic-control-plane-30.txt>>.

[I-D.ietf-anima-grasp]

Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", Work in Progress,

- Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-grasp-15.txt>>.
- [IDevID] "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [ITU.X690.1994]
International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.
- [REST] Fielding, R.F., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <<https://www.rfc-editor.org/info/rfc3927>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.

- [RFC4519] Sciberras, A., Ed., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, DOI 10.17487/RFC4519, June 2006, <<https://www.rfc-editor.org/info/rfc4519>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <<https://www.rfc-editor.org/info/rfc5272>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <<https://www.rfc-editor.org/info/rfc7469>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <<https://www.rfc-editor.org/info/rfc8368>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

13.2. Informative References

- [brewski] "Urban Dictionary: Brewski", October 2019, <<https://www.urbandictionary.com/define.php?term=brewski>>.
- [cabforumaudit] "Information for Auditors and Assessors", August 2019, <<https://cabforum.org/information-for-auditors-and-assessors/>>.
- [Dingledine2004] Dingledine, R., Mathewson, N., and P. Syverson, "Tor: the second-generation onion router", 2004, <<https://spec.torproject.org/tor-spec>>.

[dnssecroot]

"DNSSEC Practice Statement for the Root Zone ZSK Operator", December 2017,
<<https://www.iana.org/dnssec/dps/zsk-operator/dps-zsk-operator-v2.0.pdf>>.

[docsisroot]

"CableLabs Digital Certificate Issuance Service", February 2018, <<https://www.cablelabs.com/resources/digital-certificate-issuance-service/>>.

[I-D.ietf-ace-coap-est]

Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", Work in Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6 January 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-ace-coap-est-18.txt>>.

[I-D.ietf-anima-constrained-voucher]

Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-09, 2 November 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-constrained-voucher-09.txt>>.

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

Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", Work in Progress, Internet-Draft, draft-ietf-anima-reference-model-10, 22 November 2018, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-reference-model-10.txt>>.

[I-D.ietf-netconf-keystore]

Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-20, 20 August 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-netconf-keystore-20.txt>>.

[I-D.richardson-anima-state-for-joinrouter]

Richardson, M., "Considerations for stateful vs stateless join router in ANIMA bootstrap", Work in Progress, Internet-Draft, draft-richardson-anima-state-for-joinrouter-03, 22 September 2020, <<http://www.ietf.org/internet-drafts/draft-richardson-anima-state-for-joinrouter-03.txt>>.

- [imprinting] "Wikipedia article: Imprinting", July 2015, <[https://en.wikipedia.org/wiki/Imprinting_\(psychology\)](https://en.wikipedia.org/wiki/Imprinting_(psychology))>.
- [IoTstrangeThings] "IoT of toys stranger than fiction: Cybersecurity and data privacy update (accessed 2018-12-02)", March 2017, <<https://www.welivesecurity.com/2017/03/03/internet-of-things-security-privacy-iot-update/>>.
- [livingwithIoT] "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>>.
- [minerva] Richardsdon, M., "Minerva reference implementation for BRSKI", 2020, <<https://minerva.sandelman.ca/>>.
- [minervagithub] Richardsdon, M., "GITHUB hosting of Minerva reference code", 2020, <<https://github.com/ANIMAgus-minerva>>.
- [openssl] "OpenSSL X509 utility", September 2019, <<https://www.openssl.org/docs/man1.1.1/man1/openssl-x509.html>>.
- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <<https://www.rfc-editor.org/info/rfc2663>>.
- [RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <<https://www.rfc-editor.org/info/rfc5209>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<https://www.rfc-editor.org/info/rfc5785>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.

- [RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, DOI 10.17487/RFC6961, June 2013, <<https://www.rfc-editor.org/info/rfc6961>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [slowloris]
"Slowloris (computer security)", February 2019, <[https://en.wikipedia.org/wiki/Slowloris_\(computer_security\)](https://en.wikipedia.org/wiki/Slowloris_(computer_security))>.
- [softwareescrow]
"Wikipedia article: Software Escrow", October 2019, <https://en.wikipedia.org/wiki/Source_code_escrow>.
- [Stajano99theresurrecting]
Stajano, F. and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks", 1999, <<https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>>.
- [TR069] "TR-69: CPE WAN Management Protocol", February 2018, <<https://www.broadband-forum.org/standards-and-software/technical-specifications/tr-069-files-tools>>.

[W3C.WD-capability-urls-20140218]

Tennison, J., "Good Practices for Capability URLs", World Wide Web Consortium WD WD-capability-urls-20140218, 18 February 2014, <<https://www.w3.org/TR/2014/WD-capability-urls-20140218>>.

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 Pledge 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 _bootstrapks._tcp.local response doesn't work then the second and third responses are tried. If these fail the pledge moves on to normal DNS-based Service Discovery.

Appendix C. 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 reproduceable examples the public and private keys for an example MASA and registrar are first listed.

The keys come from an open source reference implementation of BRSKI, called "Minerva" [minerva]. It is available on github [minervagithub]. The keys presented here are used in the unit and integration tests. The MASA code is called "highway", the Registrar code is called "fountain", and the example client is called "reach".

The public key components of each are presented as both base64 certificates, as well as being decoded by openssl's x509 utility so that the extensions can be seen. This was version 1.1.1c of the [openssl] library and utility.

C.1. Keys involved

The Manufacturer has a Certificate Authority that signs the pledge's IDDevID. In addition the Manufacturer's signing authority (the MASA) signs the vouchers, and that certificate must be distributed to the devices at manufacturing time so that vouchers can be validated.

C.1.1.1. Manufacturer Certificate Authority for IDevID signatures

This private key is Certificate Authority that signs IDevID certificates:

```
<CODE BEGINS> file "vendor.key"
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDcAYkoLW1IEA5SKKhMMdkTK7sJxk5ybKqYq9Yr5aR7tNwqXyLGS7z8G
8S4w/UJ58BqgBwYFK4EEACKhZANiAAQu5/yktJbFLjMC87h7b+yTreFuF8GwewKH
L4mS0r0dVAQubqDUQcTrjvpXrXCpTojiLCzgp8fzkcUDkZ9LD/M90LDipiLNIokP
juF8QkoAbT8pMrY83MS8y76wZ7AalNQ=
-----END EC PRIVATE KEY-----
<CODE ENDS>
```

This public key validates IDevID certificates:

file: examples/vendor.key

```
<CODE BEGINS> file "vendor.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 519772114 (0x1efb17d2)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.ex
ample.com CA
    Validity
      Not Before: Feb 12 22:22:21 2019 GMT
      Not After : Feb 11 22:22:21 2021 GMT
    Subject: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.e
xample.com CA
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (384 bit)
      pub:
        04:2e:e7:fc:a4:b4:96:c5:2e:33:02:f3:b8:7b:6f:
        ec:93:ad:e1:6e:17:c1:b0:7b:02:87:2f:89:92:d2:
        bd:1d:54:04:2e:6e:a0:d4:41:c4:eb:8e:fa:57:ad:
        70:a9:4e:88:e2:2c:2c:e0:a7:c7:f3:91:c5:03:91:
        9f:4b:0f:f3:3d:d0:b0:e2:a6:22:cd:20:e9:0f:8e:
        e1:7c:42:4a:00:6d:3f:29:32:b6:3c:dc:c4:bc:cb:
        be:b0:67:b0:1a:94:d4
      ASN1 OID: secp384r1
      NIST CURVE: P-384
    X509v3 extensions:
      X509v3 Basic Constraints: critical
        CA:TRUE
      X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
      X509v3 Subject Key Identifier:
```

5E:0C:A9:52:5A:8C:DF:A9:0F:03:14:E9:96:F1:80:76:8C:53:8A:08
X509v3 Authority Key Identifier:
keyid:5E:0C:A9:52:5A:8C:DF:A9:0F:03:14:E9:96:F1:80:76:8C:53:8A

:08

Signature Algorithm: ecdsa-with-SHA256

30:65:02:30:5f:21:fd:c6:ab:d6:94:a6:cd:ca:37:2c:81:33:
87:fe:7b:e1:b5:1a:e8:6c:05:43:a6:8b:4e:22:b5:55:e9:48:
0c:b5:97:f3:c9:1a:65:d9:97:4b:f0:21:86:0d:cb:26:02:31:
00:e3:2d:0d:08:49:4d:a3:f5:dc:57:1f:a7:13:26:a4:e0:d6:
3a:c2:d5:4a:50:83:62:26:2e:79:2b:d0:a5:ee:66:d5:bf:16:
9a:33:75:b4:d1:8d:ba:d3:50:77:6b:92:df

-----BEGIN CERTIFICATE-----

MIICTDCCAdKgAwIBAgIEHvsX0jAKBgqhkhjOPQQDAjBdMQ8wDQYDVQQGEwZDYW5h
ZGExEDAOBgNVBAGMB09udGFyaW8xEjAQBgNVBAsMCVNhbmRlbG1hbG1jEkMCIGA1UE
AwwbAGlnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMB4XDTE5MDIxMjIyMVowX
DTIxMDIxMTIyMjIyMVowXTEPMA0GA1UEBhMGQ2FuYWRhMRAdDgYDVQQIDAdPbnRh
cm1vMRIwEAYDVQQLDAlTYW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktZGVzdC5l
eGFtcGx1LmNvbSBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABC7n/KS0lsUuMwLz
uHtv7JOt4W4XwbB7AocviZLSvR1UBC5uoNRBxOuO+letcK10i0IsL0Cnx/ORxQOR
n0sP8z3QsOKmIs0g6Q+04XxCsGbtPykytjzcxLzLvrbnsBqU1KNjMGEwDwYDVR0T
AQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVRO0BBYEFF4MqVJaJN+pDwMU
6ZbxgHaMU4oIMB8GA1UdIwQYMBaAFF4MqVJaJN+pDwMU6ZbxgHaMU4oIMAOGCCqG
SM49BAMCA2gAMGUCCMF8h/carlpSmzco3LIEzh/574bUa6GwFQ6aLTiK1VelIDLWX
88kaZdmXS/Ahhg3LJgIXAOMtDQhJTaP13FcfxMmpODWosLVS1CDYiYueSvQpe5m
1b8WmjN1tNGNutNQd2uS3w==

-----END CERTIFICATE-----

<CODE ENDS>

C.1.2. MASA key pair for voucher signatures

The MASA is the Manufacturer Authorized Signing Authority. This
keypair signs vouchers. An example TLS certificate Section 5.4 HTTP
authentication is not provided as it is a common form.

This private key signs the vouchers which are presented below:

<CODE BEGINS> file "masa.key"

-----BEGIN EC PRIVATE KEY-----

MHcCAQEEIFhdd0eDdzip67kXx72K+KHGJQYJHNY8pkiLJ6CcvxMG0AoGCCqGSM49
AwEHoUQDQgAEqgQVo0S54kT4yfkBxumdHOCrpsqbOpMKmiMln3oB1HAW25MJV+
gqi4tMFFsJ0iEwt8kszfWXX4rLgJS2mnpQ==

-----END EC PRIVATE KEY-----

<CODE ENDS>

This public key validates vouchers, and it has been signed by the CA
above:

file: examples/masa.key

C.1.3. Registrar Certificate Authority

This Certificate Authority enrolls the pledge once it is authorized, and it also signs the Registrar's certificate.

```
<CODE BEGINS> file "ownerca_secp384r1.key"
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDCHnLI0MSOLf8XndiZqoZdqblcPR5YSoPGhPOuFxFxWylgFi9HbWv8b/R
EGdRgGEVSjKgBwYFK4EEACKhZANiAAQbf1m6F8MavGaNjGzgw/oxcQ9l9iKRvbdW
gAfb37h6pUVNeYpGlxlZljGxj2l9Mr48yD5bY7VG9qjVb5v5wPPTuRQ/ckdRpHbd
0vC/9cqPMAF/+MJf0/UgA0SLi/IHbLQ=
-----END EC PRIVATE KEY-----
<CODE ENDS>
```

The public key is indicated in a pledge voucher-request to show proximity.

file: examples/ownerca_secp384r1.key

```
<CODE BEGINS> file "ownerca_secp384r1.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 694879833 (0x296b0659)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstru
ng Fountain Root CA
    Validity
      Not Before: Feb 25 21:31:45 2020 GMT
      Not After : Feb 24 21:31:45 2022 GMT
    Subject: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstr
ung Fountain Root CA
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (384 bit)
      pub:
        04:1b:7f:59:ba:17:c3:1a:bc:66:8d:8c:6c:e0:c3:
        fa:31:71:0f:65:f6:22:91:bd:b7:56:80:07:db:df:
        b8:7a:a5:45:4d:79:8a:46:97:19:59:96:31:b1:8f:
        69:7d:32:be:3c:c8:3e:5b:63:b5:46:f6:a8:d5:6f:
        9b:f9:c0:f3:d3:b9:14:3f:72:47:51:a4:76:dd:d2:
        f0:bf:f5:ca:8f:30:01:7f:f8:c2:5f:d3:f5:20:03:
        44:8b:8b:f2:07:6c:b4
      ASN1 OID: secp384r1
      NIST CURVE: P-384
    X509v3 extensions:
      X509v3 Basic Constraints: critical
      CA:TRUE
      X509v3 Key Usage: critical
      Certificate Sign, CRL Sign
```

X509v3 Subject Key Identifier:

B9:A5:F6:CB:11:E1:07:A4:49:2C:A7:08:C6:7C:10:BC:87:B3:74:26

X509v3 Authority Key Identifier:

keyid:B9:A5:F6:CB:11:E1:07:A4:49:2C:A7:08:C6:7C:10:BC:87:B3:74

:26

Signature Algorithm: ecdsa-with-SHA256

30:64:02:30:20:83:06:ce:8d:98:a4:54:7a:66:4c:4a:3a:70:
c2:52:36:5a:52:8d:59:7d:20:9b:2a:69:14:58:87:38:d8:55:
79:dd:fd:29:38:95:1e:91:93:76:b4:f5:66:29:44:b4:02:30:
6f:38:f9:af:12:ed:30:d5:85:29:7c:b1:16:58:bd:67:91:43:
c4:0d:30:f9:d8:1c:ac:2f:06:dd:bc:d5:06:42:2c:84:a2:04:
ea:02:a4:5f:17:51:26:fb:d9:2f:d2:5c

-----BEGIN CERTIFICATE-----

MIICazCCAFKgAwIBAgIEKWsGWTAKBgqhkhjOPQQDAjBtMRIwEAYKCZImizPyLGQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMMM2ZvdW50
YWluLXRlc3QuZXRhbXBsZS5jb20gVW5zdHJlbnRhaW4gUm9vdCBDQTAe
Fw0yMDAyMjUyMTMxNDVaFw0yMjUyMTMxNDVaMG0xEjAQBgoJkiaJk/IsZAEZ
FgJjYTEZMBCGcmSjOmT8ixkARkWCXNhbmRlbG1hbG1hbmRhaW4gUm9vdCBDQTAe
aW4tdGVzdC5leGFtcGxlLnNvbSBVbnN0cnVuZyBGb3VudGFpbiBSb290IENBMHYw
EAYHkoZiZjOCAQYFK4EEACIDYgAEG39ZuhfDGrxmjYxs4MP6MXEPZfYikb23VoAH
29+4eqVFTXmKRpcZWZYxsY9pfTK+PMg+W20lRvao1W+b+cDz07kUP3JHUaR23dLw
v/XKjzABf/jCX9P1IANEi4vyB2y0o2MwYTAPBgNVHRMBAf8EBTADAQH/MA4GA1Ud
DwEB/wQEAwIBBjAdBgNVHQ4EFgQUuaX2yxHhB6RjLkCIXnwQvIezdCYwHwYDVR0j
BBgwFoAUuaX2yxHhB6RjLkCIXnwQvIezdCYwCgYIKoZIzj0EAwIDZwAwZAIwIIMG
zo2YpFR6ZkxKOnDCUjZaUo1ZfSCbKmkUWic42FV53f0pOJUekZN2tPVmKUS0AjBv
OPmvEu0w1YUpfLEWWLlnkUPEDTD52BysLwbdvNUGQiyEogTqAqRfF1Em+9kv0lw=
-----END CERTIFICATE-----

<CODE ENDS>

C.1.4. Registrar key pair

The Registrar is the representative of the domain owner. This key signs registrar voucher-requests, and terminates the TLS connection from the pledge.

<CODE BEGINS> file "jrc_prime256v1.key"

-----BEGIN EC PRIVATE KEY-----

MHcCAQEIEIFZodk+PC5Mu24+ra0sbOjKzan+dW5rvDAR7YuJUOC1YoAoGCCqGSM49
AwEHoUQDQgAEIlnVQcjS6n+Xd5l/28IFv6UiegQwSBztGj5dkK2MAjQIPV8l8lH+E
jLIOYdbJiIOVtEIf1/Jqt+TOBfinTNOLog==

-----END EC PRIVATE KEY-----

<CODE ENDS>

The public key is indicated in a pledge voucher-request to show proximity.

```

<CODE BEGINS> file "jrc_prime256v1.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1066965842 (0x3f989b52)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstru
ng Fountain Root CA
    Validity
      Not Before: Feb 25 21:31:54 2020 GMT
      Not After : Feb 24 21:31:54 2022 GMT
    Subject: DC = ca, DC = sandelman, CN = fountain-test.example.com
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:96:65:50:72:34:ba:9f:e5:dd:e6:5f:f6:f0:81:
        6f:e9:48:9e:81:0c:12:07:3b:46:8f:97:64:2b:63:
        00:8d:02:0f:57:c9:7c:94:7f:84:8c:b2:0e:61:d6:
        c9:88:8d:15:b4:42:1f:d7:f2:6a:b7:e4:ce:05:f8:
        a7:4c:d3:8b:3a
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Extended Key Usage: critical
      CMC Registration Authority
      X509v3 Key Usage: critical
      Digital Signature
    Signature Algorithm: ecdsa-with-SHA256
      30:65:02:30:66:4f:60:4c:55:48:1e:96:07:f8:dd:1f:b9:c8:
      12:8d:45:36:87:9b:23:c0:bc:bb:f1:cb:3d:26:15:56:6f:5f:
      1f:bf:d5:1c:0e:6a:09:af:1b:76:97:99:19:23:fd:7e:02:31:
      00:bc:ac:c3:41:b0:ba:0d:af:52:f9:9c:6e:7a:7f:00:1d:23:
      c8:62:01:61:bc:4b:c5:c0:47:99:35:0a:0c:77:61:44:01:4a:
      07:52:70:57:00:75:ff:be:07:0e:98:cb:e5
-----BEGIN CERTIFICATE-----
MIIB/DCCAYKgAwIBAgIEP5ibUjAKBggqhkJOPQQDAjBtMRIwEAYKCZImizPyLGQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMMM2ZvdW50
YWluLXRlc3QuZXRhbXBsZS5jb20gVW5zdHJ1bmNldG91bnRhaW4gUm9vdCBDQTAe
Fw0yMDAyMjUyMTMxNTRaFw0yMjAyMjUyMTMxNTRaMFMEjAQBgoJkiaJk/IsZAEZ
FgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1hbG1hbmRhaW4gUm9vdCBDQTAe
aW4tdGVzdC5leGFtcGxlLmNvbTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABJZl
UHI0up/l3eZf9vCBb+lInoEMEGc7Ro+XZCtjAI0CD1fJfJR/hIyyDmHWyYiNFbRC
H9fyarfkzqX4p0zTizqjKjAoMBYGA1UdJQEB/wQMMAoGCCsGAQUFBwMcMA4GA1Ud
DwEB/wQEAwIHgDAKBggqhkJOPQQDAgNoADBIAjBmT2BMVUgelgf43R+5yBKNRTaH
myPAvLvxyz0mFVZvXx+/lRwOagmvG3aXmRk j/X4CMQC8rMNBsLoNr1L5nG56fwAd
I8hiAWG8S8XAR5k1Cgx3YUQBSgdScFcAdf++Bw6Yy+U=
-----END CERTIFICATE-----
<CODE ENDS>

```

C.1.5. Pledge key pair

The pledge has an IDevID key pair built in at manufacturing time:

```
<CODE BEGINS> file "idevid_00-D0-E5-F2-00-02.key"
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBHNh6r8QRevRuo+tEmBJeFjQKf6bpFA/9NGoltv+9sNoAoGCCqGSM49
AwEHoUQDQgAEA6N1Q4ezfMAKmoecrfb00BMclAyEH+BATkF58FsTSyBxs0SbSWLx
FjDOuWB9gLGn2TsTUJumJ6VPw5Z/TP4hJw==
-----END EC PRIVATE KEY-----
<CODE ENDS>
```

The certificate is used by the registrar to find the MASA.

```
<CODE BEGINS> file "idevid_00-D0-E5-F2-00-02.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 226876461 (0xd85dc2d)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.ex
ample.com CA
    Validity
      Not Before: Feb  3 06:47:20 2020 GMT
      Not After : Dec 31 00:00:00 2999 GMT
    Subject: serialNumber = 00-D0-E5-F2-00-02
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:03:a3:75:43:87:b3:7c:c0:0a:9a:87:9c:ad:f6:
        f4:38:13:1c:d4:0c:84:1f:e0:40:4e:41:79:f0:5b:
        13:4b:20:71:b3:44:9b:49:62:f1:16:30:ce:bb:00:
        7d:80:b1:a7:d9:3b:13:50:9b:a6:27:a5:4f:c3:96:
        7f:4c:fe:21:27
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        45:88:CC:96:96:00:64:37:B0:BA:23:65:64:64:54:08:06:6C:56:AD
      X509v3 Basic Constraints:
        CA:FALSE
      1.3.6.1.5.5.7.1.32:
        ..highway-test.example.com:9443
    Signature Algorithm: ecdsa-with-SHA256
      30:65:02:30:23:e1:a9:2e:ef:22:12:34:5a:a5:c2:15:d6:28:
      bd:ed:3d:96:d6:ce:04:95:ef:a7:c8:dc:18:a8:31:c7:b8:04:
      34:f2:b7:4d:79:8a:67:22:24:03:4f:c5:cd:d6:06:ba:02:31:
      00:b3:8d:5c:0a:d0:fe:04:83:90:d3:4f:6d:72:97:b3:3e:02:
```

```

        ea:f1:c8:5a:32:72:58:b7:45:02:50:78:bc:04:1d:23:5e:22:
        6f:c3:7f:8c:7c:d7:9b:70:20:91:b4:e1:7f
-----BEGIN CERTIFICATE-----
MIIB5jCCAWygAwIBAgIEDYXcLTAKBggqhkJOPQQDAjBdMQ8wDQYDVQQGEwZDYW5h
ZGExEDAOBgNVBAGMB09udGFyaW8xEjAQBgNVBAsMCVNhbmRlbG1hbJEkMCIGA1UE
AwwbaglnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMCAXDTEwMDIwMzA2NDcyMFoY
DzI5OTkxMjMxMDAwMDAwWjAcMRowGAYDVQQFDBEwMC1EMC1FNS1GMi0wMC0wMjBZ
MBMGBYqGSM49AgEGCCqGSM49AwEHA0IABAOjdUOHs3zACpqHnK329DgTHNQMb/g
QE5BefBbE0sgcbNEm0li8RYwzrsAfYCYxp9k7E1CbpielT8OWf0z+ISejWTBXMB0G
A1UdDgQWBRRFiMyWlgBkN7C6I2VkZFQIBmxWrTAJBgNVHRMEAjAAMCsGCCsGAQUF
BwEgBB8MHWhpZ2h3YXktdGVzdC5leGFtcGx1LmNvbTo5NDQzMAoGCCqGSM49BAMC
A2gAMGUCMCPHqS7vIhI0WqXCFdYove09ltbOBjXvp8jcGKgxx7gENPK3TXmKzyIk
A0/FzdYGugIxALONXArQ/gSDkNNPbXKXsz4C6vHIWjJyWLdFA1B4vAQdI14ib8N/
jHzXm3AgkbThfw==
-----END CERTIFICATE-----
<CODE ENDS>

```

C.2. Example process

The JSON examples below are wrapped at 60 columns. This results in strings that have newlines in them, which makes them invalid JSON as is. The strings would otherwise be too long, so they need to be unwrapped before processing.

For readability, the output of the `asn1parse` has been truncated at 72 columns rather than wrapped.

C.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-registrar-cert field. The base64 has been wrapped at 60 characters for presentation reasons.

```
<CODE BEGINS> file "vr_00-D0-E5-F2-00-02.b64"
MIIG3wYJKoZIhvcNAQcCoIIIG0DCCBswCAQEExDTALBg1ghkgBZQMEAgEwggOJBgkqhkiG9w0BBwGg
ggN6BIIDdnsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2hlciI6eyJhc3NlcnRpb24iOiJwcm94
aWlpdHkiLCJjcmVhdGVkLW9uIjoimjAyMC0wMi0yNVQxODowND00C42NTItMDU6MDAiLCJzZXJp
YWwtbnVtYmVyIjoimDAtRDAtRTUtRjItMDAtMDIiLCJub25jZSI6ImFNamd1ZUtVVC0yMndWaWlq
NnoyN1EiLCJwcm94aWlpdHktcmVnaXN0cmFyLWNlcnQiOiJNSU1CL0RDQ0FZS2dBd0lCQWdJRVA1
aWJVakFLQmdncWhrak9QUVFEQWpCdE1SSXdfQVlLQ1pJbWlaUH1MR1FCR1JZQ1kyRXhhHVEFYQmdv
SmtpYUprL01zWkFFWkZnbHpZVzVrWld4dFlXNHhQREE2QmdOVkJBtU1NM1p2ZFclMF1XbHVMWFJs
YzNRdVpYaGhiWEJzWlM1amIyMGdWVzV6ZEhKMWJtY2dSbTkxYm5SaGFxNGdVbTl2ZENCRFFUQWVG
dzB5TURBeU1qVXlNVE14TlRSYUZ3MH1NakF5TWpReU1UTXhOVFJhTUZNeEVqQVFCZ29Ka2lhSmsv
SXNaQUVaRmdKallURVpNQMNHQ2dtU0pvbVQ4aXhrQVJrV0NYTmhibVJsYkcxagJqRW1NQ0FHQTFV
RUF3dlpabTkxYm5SaGFxNHRkR1Z6ZEM1bGVHRnRjR3hsTG1OdmJUQlpNQk1HQnlxR1NNND1BZ0VH
QONxR1NNND1Bd0VlQTBjQUUJKWmxVSEkwdXAvbDN1WmY5dkNCYitsSW5vRU1FZ2M3Um8rWFpDdGpB
STBDRDFmSmZKU19oSX15RG1IV31ZaU5GY1JDSdlmeWFyZmt6Z1g0cDB6VG16cWpLakFvTUJZR0Ex
VWRKUUVCL3dRTU1Bb0dDQ3NHQVFVRkRjJ3TWNNTQTRHQTfVZER3RUIvd1FFQXdxJSGdEQUtCZ2dxaGtq
T1BRUURBZ05vQURCbEFqQm1UMk1JNV1VnZWxnZjQzUis1eUJLTlJUyUhteVBBdkx2eHl6MG1GVlp2
WHgrLzFSd09hZ212RzNhWG1Sa2ovWDRDRTVfDOHJNTkZjZTG9OcJFMNW5HNTZmd0FkSThoaUFXRzhT
OFhBUjVrMUNneDNZVVFcu2dkU2NGY0FkZisrQnc2WXkrVT0ifX2gggHqMIIB5jCCAWygAwIBAgIE
DYXcLTAKBggqhkiJOPQDAjBdMQ8wDQYDVQQGEWZDYW5hZGExEDAObGVBAgMB09udGFyaW8xEjAQ
BgNVBAAsMCVhbmRlbG1hbG1hbmRlZGE1UEAwbaG1naHdheS10ZXN0LmV4YW1wbGUuY29tIENBMCAx
DTIwMDIwMzA2NDcyMFOYDzI5OTkxMjMxMDAwMDAwWjAcMR0wGAYDVQQFDBEwMC1EMC1FNS1GMi0w
MC0wMjBZMBMBGByqGSM49AgEGCCqGSM49AwEHA0IABAOjdUOHs3ZACpqHnK329DgTHNQMHb/gQE5B
efBbE0sgcbNEM0li8RYwzrsAfYCx9k7E1CbpielT8OWf0z+ISejWTBxMB0GA1UdDgQWBBrFiMyW
lgBkN7C6I2VkJBmXWrtAJBgNVHRMEAjaAMCsGCCsGAQUFBwEgBB8MHWhpZ2h3YXktZGVzdC5l
eGftcGx1LmNvbTo5NDQzMAoGCCqGSM49BAMCA2gAMGUCMCPHqs7vIhI0WqXCFdYove091tbOBjXv
p8jcGKgx7gENPK3TXmKZyIkA0/FzdYGugIxALONXArQ/gSDkNNPbXKXsZ4C6vHIWjJyWldFA1B4
vAQdI14ib8N/jHzXm3AgkbThfzGCATswggE3AgEBMGUwXTEPMA0GA1UEBhMGQ2FuYWRhMRAdGyYD
VQQIDAdPbnRhcmlvMRIwEAYDVQQQLDAlTYW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktZGVzdC5l
eGftcGx1LmNvbSBDQIIEYXcLTALBg1ghkgBZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcN
AQcBMBwGCSqGSIb3DQEJBTEPFw0yMDAyMjUyMzA0NDhaMC8GCSqGSIb3DQEJBDEiBCCx6IrwstHF
609Y0EqDK62QKby4duyyIWudvs15M16BBTAKBggqhkiJOPQDAgRHMEUCIBxwA1U1kIkuQDf/j7kZ
/MVefgr141+hKBFgrnNngjwpAiEAy8aXt0GSB9m1bmiEUpefCEhxSv2xLYurGlugv0dfr/E=
<CODE ENDS>
```

The ASN1 decoding of the artifact:

file: examples/vr_00-D0-E5-F2-00-02.b64

```
0:d=0  hl=4  l=1759  cons: SEQUENCE
4:d=1  hl=2  l=   9  prim: OBJECT               :pkcs7-signedData
15:d=1  hl=4  l=1744  cons: cont [ 0 ]
19:d=2  hl=4  l=1740  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= 905  cons: SEQUENCE
45:d=4  hl=2  l=   9  prim: OBJECT               :pkcs7-data
```

```

56:d=4  hl=4  l= 890 cons: cont [ 0 ]
60:d=5  hl=4  l= 886 prim: OCTET STRING      :{"ietf-voucher-request:v
950:d=3  hl=4  l= 490 cons: cont [ 0 ]
954:d=4  hl=4  l= 486 cons: SEQUENCE
958:d=5  hl=4  l= 364 cons: SEQUENCE
962:d=6  hl=2  l=   3 cons: cont [ 0 ]
964:d=7  hl=2  l=   1 prim: INTEGER           :02
967:d=6  hl=2  l=   4 prim: INTEGER           :0D85DC2D
973:d=6  hl=2  l=  10 cons: SEQUENCE
975:d=7  hl=2  l=   8 prim: OBJECT            :ecdsa-with-SHA256
985:d=6  hl=2  l=  93 cons: SEQUENCE
987:d=7  hl=2  l=  15 cons: SET
989:d=8  hl=2  l=  13 cons: SEQUENCE
991:d=9  hl=2  l=   3 prim: OBJECT            :countryName
996:d=9  hl=2  l=   6 prim: PRINTABLESTRING   :Canada
1004:d=7  hl=2  l=  16 cons: SET
1006:d=8  hl=2  l=  14 cons: SEQUENCE
1008:d=9  hl=2  l=   3 prim: OBJECT            :stateOrProvinceName
1013:d=9  hl=2  l=   7 prim: UTF8STRING       :Ontario
1022:d=7  hl=2  l=  18 cons: SET
1024:d=8  hl=2  l=  16 cons: SEQUENCE
1026:d=9  hl=2  l=   3 prim: OBJECT            :organizationalUnitName
1031:d=9  hl=2  l=   9 prim: UTF8STRING       :Sandelman
1042:d=7  hl=2  l=  36 cons: SET
1044:d=8  hl=2  l=  34 cons: SEQUENCE
1046:d=9  hl=2  l=   3 prim: OBJECT            :commonName
1051:d=9  hl=2  l=  27 prim: UTF8STRING       :highway-test.example.com
1080:d=6  hl=2  l=  32 cons: SEQUENCE
1082:d=7  hl=2  l=  13 prim: UTCTIME          :200203064720Z
1097:d=7  hl=2  l=  15 prim: GENERALIZEDTIME  :29991231000000Z
1114:d=6  hl=2  l=  28 cons: SEQUENCE
1116:d=7  hl=2  l=  26 cons: SET
1118:d=8  hl=2  l=  24 cons: SEQUENCE
1120:d=9  hl=2  l=   3 prim: OBJECT            :serialNumber
1125:d=9  hl=2  l=  17 prim: UTF8STRING       :00-D0-E5-F2-00-02
1144:d=6  hl=2  l=  89 cons: SEQUENCE
1146:d=7  hl=2  l=  19 cons: SEQUENCE
1148:d=8  hl=2  l=   7 prim: OBJECT            :id-ecPublicKey
1157:d=8  hl=2  l=   8 prim: OBJECT            :prime256v1
1167:d=7  hl=2  l=  66 prim: BIT STRING
1235:d=6  hl=2  l=  89 cons: cont [ 3 ]
1237:d=7  hl=2  l=  87 cons: SEQUENCE
1239:d=8  hl=2  l=  29 cons: SEQUENCE
1241:d=9  hl=2  l=   3 prim: OBJECT            :X509v3 Subject Key Ident
1246:d=9  hl=2  l=  22 prim: OCTET STRING     [HEX DUMP]:04144588CC9696
1270:d=8  hl=2  l=   9 cons: SEQUENCE
1272:d=9  hl=2  l=   3 prim: OBJECT            :X509v3 Basic Constraints
1277:d=9  hl=2  l=   2 prim: OCTET STRING     [HEX DUMP]:3000

```

```

1281:d=8  hl=2 l= 43 cons: SEQUENCE
1283:d=9  hl=2 l=  8 prim: OBJECT           :1.3.6.1.5.5.7.1.32
1293:d=9  hl=2 l= 31 prim: OCTET STRING      [HEX DUMP]:0C1D6869676877
1326:d=5  hl=2 l= 10 cons: SEQUENCE
1328:d=6  hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
1338:d=5  hl=2 l=104 prim: BIT STRING
1444:d=3  hl=4 l=315 cons: SET
1448:d=4  hl=4 l=311 cons: SEQUENCE
1452:d=5  hl=2 l=  1 prim: INTEGER           :01
1455:d=5  hl=2 l=101 cons: SEQUENCE
1457:d=6  hl=2 l= 93 cons: SEQUENCE
1459:d=7  hl=2 l= 15 cons: SET
1461:d=8  hl=2 l= 13 cons: SEQUENCE
1463:d=9  hl=2 l=  3 prim: OBJECT           :countryName
1468:d=9  hl=2 l=  6 prim: PRINTABLESTRING   :Canada
1476:d=7  hl=2 l= 16 cons: SET
1478:d=8  hl=2 l= 14 cons: SEQUENCE
1480:d=9  hl=2 l=  3 prim: OBJECT           :stateOrProvinceName
1485:d=9  hl=2 l=  7 prim: UTF8STRING        :Ontario
1494:d=7  hl=2 l= 18 cons: SET
1496:d=8  hl=2 l= 16 cons: SEQUENCE
1498:d=9  hl=2 l=  3 prim: OBJECT           :organizationalUnitName
1503:d=9  hl=2 l=  9 prim: UTF8STRING        :Sandelman
1514:d=7  hl=2 l= 36 cons: SET
1516:d=8  hl=2 l= 34 cons: SEQUENCE
1518:d=9  hl=2 l=  3 prim: OBJECT           :commonName
1523:d=9  hl=2 l= 27 prim: UTF8STRING        :highway-test.example.com
1552:d=6  hl=2 l=  4 prim: INTEGER           :0D85DC2D
1558:d=5  hl=2 l= 11 cons: SEQUENCE
1560:d=6  hl=2 l=  9 prim: OBJECT           :sha256
1571:d=5  hl=2 l=105 cons: cont [ 0 ]
1573:d=6  hl=2 l= 24 cons: SEQUENCE
1575:d=7  hl=2 l=  9 prim: OBJECT           :contentType
1586:d=7  hl=2 l= 11 cons: SET
1588:d=8  hl=2 l=  9 prim: OBJECT           :pkcs7-data
1599:d=6  hl=2 l= 28 cons: SEQUENCE
1601:d=7  hl=2 l=  9 prim: OBJECT           :signingTime
1612:d=7  hl=2 l= 15 cons: SET
1614:d=8  hl=2 l= 13 prim: UTCTIME           :200225230448Z
1629:d=6  hl=2 l= 47 cons: SEQUENCE
1631:d=7  hl=2 l=  9 prim: OBJECT           :messageDigest
1642:d=7  hl=2 l= 34 cons: SET
1644:d=8  hl=2 l= 32 prim: OCTET STRING      [HEX DUMP]:B1E88AF0B2D1C5
1678:d=5  hl=2 l= 10 cons: SEQUENCE
1680:d=6  hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
1690:d=5  hl=2 l= 71 prim: OCTET STRING      [HEX DUMP]:304502201C7003

```

The JSON contained in the voucher request:

U2EyB3ZXRFJEVfZGRE9ISk5Ua0p6VEc5T2NqRk1OVzVITlRabWQwRmtTVGhvYVVGWFJ6aFRPRmhC
VWpWck1Vtm5lRE5aVlZGQ1UyZGtVMk5HWTBGa1ppc3JRbmMyV1hrc1ZUMGlmWDJnZ2dIcU1JSUI1
akNDQVd5Z0F3SUJBZ0lFRFlYY0xUQUtCZ2dxaGtqT1BRUURBakJkTVE4d0RRWURWUVFHRXdaRFlX
NWhaR0V4RURBT0JnTlZCQWdNQjA5dWRHRnlhVzh4RWpBUUJnTlZCQXNNQ1ZOaGJtUmxiRzFoYmpF
a01DSUdBMVVFQXd3YmFhbG5hSGRoZVMxMFpYtJbMbVY0WVcx2JHVXVZMj10SUVOQk1DQVhEVEl3
TURJd0l6QTJORG5TUZvWUR6STVPVGt4TWpNeE1EQXdNREF3V2pBY01Sb3dHQVlEVlFRRkRCRXdN
QzFFtUMxRk5TMudNaTB3TUMwd0lqQlpNQk1HqnlxR1NNND1BZ0VHQ0Nxr1NNND1Bd0VIQTBjQUJB
T2pkVU9IczN6QUNwcUhuSzMjY0URnVEhOUU1oQ19nUUU1QmVmQmJFMHNNY2JORW0wbGk4U1l3enJz
QWZZQ3hwOWs3RTFDYnBpZWxUOE9XZjB6K0lTZWpXVEJYtU1wR0ExVWREZlFXQkJSRmlNeVdsZ0Jr
TjdDNkkyVmtaRlFJQm14V3JUQUpCZ05WSFJNRUFqQUFNQ3NHQ0NzR0FRVUZCd0VnQkI4TUhXaHBA
MmgzWVhRdGRHVnpkQzVsZUdGdGNHeGxMbU52YlRvNU5EUXpNQW9HQ0Nxr1NNND1CQU1DQJnQU1H
VUNNQ1BocVM3dkloSTBXcVhDRmRZb3ZlMDlSDGJpQkpYdnA4amNHS2d4eDdnRU5QSzNUWGlLWnlJ
a0EwL0Z6ZFlHdWdJeEFMT05YQXJRL2dTRGtOTlBiWETYc3o0QzZ2SElXakp5V0xkRkFsQjR2QVfk
STE0aWI4T19qSHpYbTNBZ2tiVGhmekdDQVRzd2dnRTNBZ0VCTUdVd1hURVBNQTBHQTFVRUJ0TUdR
MkZlWVdSaE1SQXdEZl1lEVlFRSURBZFBib1JoY21sdK1SSXdfQVlEVlFRTERBbFRZVzVrWld4dFlX
NHhKREFPQmdOVkBTU1HmhwWjJoM1lYa3RkRlZ6ZEMlbgVHRnRjR3hsTG10dmJTQkRRU1FRFlY
Y0xUQUxXZ2xnaGtnQlpRTUVBZ0dnYVRBWUJna3Foa2lHOXcwQkNRTXhDdl1KS29aSWH2Y05BUWNC
TUJ3R0NTcUdTSWIZRFFFSkRURVBGdzB5TURBeU1qVXlNekEwTkRoYU1DOEdDU3FHU0liM0RRRUUpC
REVpQkNDdDZJcndzdEhGNjA5WTBfcURLNjRS2J5NGR1eXlJV3VkdNMXNU0xNkJCVEFLQmdncWhr
ak9QUVFEQWdSSE1FVUNJQnh3QTFVbGtJa3VRRGYvaJdrW19NvmVmZ3IxNDEraEtCRmdybk5uZ2p3
cEFpRUF50GFYdDBHU0I5bTFibWlFVXB1ZkNfahHtdjJ4TF1lckdsdWd2MGRmci9FPSJ9faCCBG8w
ggH8MIIBGqADAgEAgQ/mJtSMAoGCCqGSM49BAMCMG0xEJAQBgOJkiaJk/IsZAEZFgJjYTEZMBcG
CgmSjomT8ixkARKWCXNhbmlRlbg1hbJbE8MDoGA1UEAwZm91bnRhaW4tdGVzdC5leGftcGx1LmNv
bSBVbnN0cnVuZyBGB3VudGFpbiBSb290IENBMB4XDTEwMDIyNTIxMzE1NFoXDTEwMDIyNDIxMzE1
NFowUzESMBAGCgmSjomT8ixkARKWAmNhMRkWFwYKZCImiZPyLGQBGRIYJc2FuzGVsbWFWuMSIwIAYD
VQDDBlmb3VudGFpbi10ZXN0LmV4YW1wbGUuY29tMEFkwEwYHkoZlZjOCAQYIKoZlZjOQACQDQgAE
lmVQcjs6n+Xd5l/28IFv6UiegQwSBztGj5dkK2MAjQIPV8l8lH+EjLIOYdbJiI0VtEif1/Jqt+TO
BfinTNOLOqMqMCgWfGgYDVR0LAQH/BAwwCgYIKwYBBQUHAXwDgYDVR0PAQH/BAQDAgeAMAoGCCqG
SM49BAMCA2gAMGUCMGZPYExVSB6WB/jdH7nIEo1FNoebI8C8u/HLPsYVVM9fh7/VHA5qCa8bdpeZ
GSP9fgIXALysw0Gwug2vUvmcbnp/AB0jyGIBYbxLxcBHmTUKDHdhRAFKB1JwVwB1/74HDpjL5TCC
AmswggHyoAMCAQICBclRBlkwCgYIKoZlZjOEAwIwbTESMBAGCgmSjomT8ixkARKWAmNhMRkWFwYK
CZImiZPyLGQBGRIYJc2FuzGVsbWFWuMTwwOgYDVR0QDDNmb3VudGFpbi10ZXN0LmV4YW1wbGUuY29t
IFVuc3RydW5nIEZvdW50YWluIFJvb3QgQ0EwHhcNMjAwMjI1MjEzMTQ1WhcNMjAwMjI1MjEzMTQ1
WjBtMRIwEAYKZCImiZPyLGQBGRIYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNV
BAMMM2ZvdW50YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJlbnRhaW4gUm9vdCBDQTB2
MBAGByqGSM49AgEGBSuBBAAiA2IABBT/WboXwxq8Zo2MbODD+jFx2D2X2IpG9t1aAB9vfuHq1RU15
ikaXGVmWmBGPax0yvjzIPltjtUb2qNVvm/nA89O5FD9yR1Gkdt3S8L/1yo8wAX/4w1/T9SADRIuL
8gdstKNjMGEwDwYDVR0TAQH/BAUwAwEB/zA0BgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFml9ssR
4QekSSSynCMZ8ELyHs3QmMB8GA1UdIwQYMBaAFml9ssR4QekSSSynCMZ8ELyHs3QmMAoGCCqGSM49
BAMCA2cAMGQCMCCDBs6NmKRUemZMSjpwWlI2WlKNWX0gmypFFiHONhVed39KtiVhPGtdrT1Zile
tAIwbzj5rxLtMNWFKXyxFli9Z5FDxA0w+dgcrC8G3bzVBkIshKIE6gKkXxdRjvZL9JcMYIBSzcC
AUcCAQEwdTBTMRIwEAYKZCImiZPyLGQBGRIYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4x
PDA6BgNVBAMMM2ZvdW50YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJlbnRhaW4gUm9v
dCBDQDQIEP5ibUjALBg1ghkgBZQMEAgGaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcCBMBwGCSqG
SIb3DQEJJBTEPFw0yMDAyMjUyMzA0NDlaMC8GCSqGSIb3DQEJJBDEiBCA9gYXR1sS0giII3PwvOK/N
5RUBWjSL/cDcrH/Bd+ElaJAKBgqhkiG9w0BCQDAgRHMEUCIFieXZa07P9eZMpCvN2laB4czw7I0s0P
s9+frCJtEBTTAiEAhCcB//qmgqcEA+90mquvVNENmFH9dxCH8Ihhz6SCVDI=
<CODE ENDS>

The ASN1 decoding of the artifact:

file: examples/parboiled_vr_00_D0-E5-02-00-2D.b64

```
0:d=0  hl=4  l=4087  cons: SEQUENCE
4:d=1  hl=2  l=  9  prim: OBJECT           :pkcs7-signedData
15:d=1  hl=4  l=4072  cons: cont [ 0 ]
19:d=2  hl=4  l=4068  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=2572  cons: SEQUENCE
45:d=4  hl=2  l=  9  prim: OBJECT           :pkcs7-data
56:d=4  hl=4  l=2557  cons: cont [ 0 ]
60:d=5  hl=4  l=2553  prim: OCTET STRING      :{"ietf-voucher-request:v
2617:d=3 hl=4  l=1135  cons: cont [ 0 ]
2621:d=4 hl=4  l= 508  cons: SEQUENCE
2625:d=5 hl=4  l= 386  cons: SEQUENCE
2629:d=6 hl=2  l=  3  cons: cont [ 0 ]
2631:d=7 hl=2  l=  1  prim: INTEGER           :02
2634:d=6 hl=2  l=  4  prim: INTEGER           :3F989B52
2640:d=6 hl=2  l= 10  cons: SEQUENCE
2642:d=7 hl=2  l=  8  prim: OBJECT           :ecdsa-with-SHA256
2652:d=6 hl=2  l= 109  cons: SEQUENCE
2654:d=7 hl=2  l= 18  cons: SET
2656:d=8 hl=2  l= 16  cons: SEQUENCE
2658:d=9 hl=2  l= 10  prim: OBJECT           :domainComponent
2670:d=9 hl=2  l=  2  prim: IA5STRING        :ca
2674:d=7 hl=2  l= 25  cons: SET
2676:d=8 hl=2  l= 23  cons: SEQUENCE
2678:d=9 hl=2  l= 10  prim: OBJECT           :domainComponent
2690:d=9 hl=2  l=  9  prim: IA5STRING        :sandelman
2701:d=7 hl=2  l= 60  cons: SET
2703:d=8 hl=2  l= 58  cons: SEQUENCE
2705:d=9 hl=2  l=  3  prim: OBJECT           :commonName
2710:d=9 hl=2  l= 51  prim: UTF8STRING        :fountain-test.example.co
2763:d=6 hl=2  l= 30  cons: SEQUENCE
2765:d=7 hl=2  l= 13  prim: UTCTIME           :200225213154Z
2780:d=7 hl=2  l= 13  prim: UTCTIME           :220224213154Z
2795:d=6 hl=2  l= 83  cons: SEQUENCE
2797:d=7 hl=2  l= 18  cons: SET
2799:d=8 hl=2  l= 16  cons: SEQUENCE
2801:d=9 hl=2  l= 10  prim: OBJECT           :domainComponent
2813:d=9 hl=2  l=  2  prim: IA5STRING        :ca
2817:d=7 hl=2  l= 25  cons: SET
2819:d=8 hl=2  l= 23  cons: SEQUENCE
2821:d=9 hl=2  l= 10  prim: OBJECT           :domainComponent
```

```

2833:d=9  hl=2 l= 9 prim: IA5STRING      :sandelman
2844:d=7  hl=2 l= 34 cons: SET
2846:d=8  hl=2 l= 32 cons: SEQUENCE
2848:d=9  hl=2 l= 3 prim: OBJECT          :commonName
2853:d=9  hl=2 l= 25 prim: UTF8STRING     :fountain-test.example.co
2880:d=6  hl=2 l= 89 cons: SEQUENCE
2882:d=7  hl=2 l= 19 cons: SEQUENCE
2884:d=8  hl=2 l= 7 prim: OBJECT           :id-ecPublicKey
2893:d=8  hl=2 l= 8 prim: OBJECT           :prime256v1
2903:d=7  hl=2 l= 66 prim: BIT STRING
2971:d=6  hl=2 l= 42 cons: cont [ 3 ]
2973:d=7  hl=2 l= 40 cons: SEQUENCE
2975:d=8  hl=2 l= 22 cons: SEQUENCE
2977:d=9  hl=2 l= 3 prim: OBJECT           :X509v3 Extended Key Usag
2982:d=9  hl=2 l= 1 prim: BOOLEAN         :255
2985:d=9  hl=2 l= 12 prim: OCTET STRING   [HEX DUMP]:300A06082B0601
2999:d=8  hl=2 l= 14 cons: SEQUENCE
3001:d=9  hl=2 l= 3 prim: OBJECT           :X509v3 Key Usage
3006:d=9  hl=2 l= 1 prim: BOOLEAN         :255
3009:d=9  hl=2 l= 4 prim: OCTET STRING   [HEX DUMP]:03020780
3015:d=5  hl=2 l= 10 cons: SEQUENCE
3017:d=6  hl=2 l= 8 prim: OBJECT           :ecdsa-with-SHA256
3027:d=5  hl=2 l= 104 prim: BIT STRING
3133:d=4  hl=4 l= 619 cons: SEQUENCE
3137:d=5  hl=4 l= 498 cons: SEQUENCE
3141:d=6  hl=2 l= 3 cons: cont [ 0 ]
3143:d=7  hl=2 l= 1 prim: INTEGER         :02
3146:d=6  hl=2 l= 4 prim: INTEGER         :296B0659
3152:d=6  hl=2 l= 10 cons: SEQUENCE
3154:d=7  hl=2 l= 8 prim: OBJECT           :ecdsa-with-SHA256
3164:d=6  hl=2 l= 109 cons: SEQUENCE
3166:d=7  hl=2 l= 18 cons: SET
3168:d=8  hl=2 l= 16 cons: SEQUENCE
3170:d=9  hl=2 l= 10 prim: OBJECT          :domainComponent
3182:d=9  hl=2 l= 2 prim: IA5STRING       :ca
3186:d=7  hl=2 l= 25 cons: SET
3188:d=8  hl=2 l= 23 cons: SEQUENCE
3190:d=9  hl=2 l= 10 prim: OBJECT          :domainComponent
3202:d=9  hl=2 l= 9 prim: IA5STRING       :sandelman
3213:d=7  hl=2 l= 60 cons: SET
3215:d=8  hl=2 l= 58 cons: SEQUENCE
3217:d=9  hl=2 l= 3 prim: OBJECT          :commonName
3222:d=9  hl=2 l= 51 prim: UTF8STRING     :fountain-test.example.co
3275:d=6  hl=2 l= 30 cons: SEQUENCE
3277:d=7  hl=2 l= 13 prim: UTCTIME        :200225213145Z
3292:d=7  hl=2 l= 13 prim: UTCTIME        :220224213145Z
3307:d=6  hl=2 l= 109 cons: SEQUENCE
3309:d=7  hl=2 l= 18 cons: SET

```

```

3311:d=8  hl=2 l= 16 cons: SEQUENCE
3313:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3325:d=9  hl=2 l=  2 prim: IA5STRING        :ca
3329:d=7  hl=2 l= 25 cons: SET
3331:d=8  hl=2 l= 23 cons: SEQUENCE
3333:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3345:d=9  hl=2 l=  9 prim: IA5STRING        :sandelman
3356:d=7  hl=2 l= 60 cons: SET
3358:d=8  hl=2 l= 58 cons: SEQUENCE
3360:d=9  hl=2 l=  3 prim: OBJECT           :commonName
3365:d=9  hl=2 l= 51 prim: UTF8STRING       :fountain-test.example.co
3418:d=6  hl=2 l= 118 cons: SEQUENCE
3420:d=7  hl=2 l= 16 cons: SEQUENCE
3422:d=8  hl=2 l=  7 prim: OBJECT           :id-ecPublicKey
3431:d=8  hl=2 l=  5 prim: OBJECT           :secp384r1
3438:d=7  hl=2 l= 98 prim: BIT STRING
3538:d=6  hl=2 l= 99 cons: cont [ 3 ]
3540:d=7  hl=2 l= 97 cons: SEQUENCE
3542:d=8  hl=2 l= 15 cons: SEQUENCE
3544:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Basic Constraints
3549:d=9  hl=2 l=  1 prim: BOOLEAN          :255
3552:d=9  hl=2 l=  5 prim: OCTET STRING     [HEX DUMP]:30030101FF
3559:d=8  hl=2 l= 14 cons: SEQUENCE
3561:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Key Usage
3566:d=9  hl=2 l=  1 prim: BOOLEAN          :255
3569:d=9  hl=2 l=  4 prim: OCTET STRING     [HEX DUMP]:03020106
3575:d=8  hl=2 l= 29 cons: SEQUENCE
3577:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Subject Key Ident
3582:d=9  hl=2 l= 22 prim: OCTET STRING     [HEX DUMP]:0414B9A5F6CB11
3606:d=8  hl=2 l= 31 cons: SEQUENCE
3608:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Authority Key Ide
3613:d=9  hl=2 l= 24 prim: OCTET STRING     [HEX DUMP]:30168014B9A5F6
3639:d=5  hl=2 l= 10 cons: SEQUENCE
3641:d=6  hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
3651:d=5  hl=2 l= 103 prim: BIT STRING
3756:d=3  hl=4 l= 331 cons: SET
3760:d=4  hl=4 l= 327 cons: SEQUENCE
3764:d=5  hl=2 l=  1 prim: INTEGER          :01
3767:d=5  hl=2 l= 117 cons: SEQUENCE
3769:d=6  hl=2 l= 109 cons: SEQUENCE
3771:d=7  hl=2 l= 18 cons: SET
3773:d=8  hl=2 l= 16 cons: SEQUENCE
3775:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3787:d=9  hl=2 l=  2 prim: IA5STRING        :ca
3791:d=7  hl=2 l= 25 cons: SET
3793:d=8  hl=2 l= 23 cons: SEQUENCE
3795:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3807:d=9  hl=2 l=  9 prim: IA5STRING        :sandelman

```

```
3818:d=7  hl=2 l= 60 cons: SET
3820:d=8  hl=2 l= 58 cons: SEQUENCE
3822:d=9  hl=2 l=  3 prim: OBJECT           :commonName
3827:d=9  hl=2 l= 51 prim: UTF8STRING       :fountain-test.example.co
3880:d=6  hl=2 l=  4 prim: INTEGER          :3F989B52
3886:d=5  hl=2 l= 11 cons: SEQUENCE
3888:d=6  hl=2 l=  9 prim: OBJECT           :sha256
3899:d=5  hl=2 l= 105 cons: cont [ 0 ]
3901:d=6  hl=2 l= 24 cons: SEQUENCE
3903:d=7  hl=2 l=  9 prim: OBJECT           :contentType
3914:d=7  hl=2 l= 11 cons: SET
3916:d=8  hl=2 l=  9 prim: OBJECT           :pkcs7-data
3927:d=6  hl=2 l= 28 cons: SEQUENCE
3929:d=7  hl=2 l=  9 prim: OBJECT           :signingTime
3940:d=7  hl=2 l= 15 cons: SET
3942:d=8  hl=2 l= 13 prim: UTCTIME          :200225230449Z
3957:d=6  hl=2 l= 47 cons: SEQUENCE
3959:d=7  hl=2 l=  9 prim: OBJECT           :messageDigest
3970:d=7  hl=2 l= 34 cons: SET
3972:d=8  hl=2 l= 32 prim: OCTET STRING     [HEX DUMP]:3D818C51D6C4B4
4006:d=5  hl=2 l= 10 cons: SEQUENCE
4008:d=6  hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
4018:d=5  hl=2 l= 71 prim: OCTET STRING     [HEX DUMP]:30450220589E5D
```

The JSON contained in the voucher request. Note that the previous voucher request is in the prior-signed-voucher-request attribute.


```
<CODE BEGINS> file "voucher_00-D0-E5-F2-00-02.b64"
MIIGxwYJKoZIhvcNAQcCoIIGuDCCBBrQCAQEhDTALBg1ghkgBZQMEAgEwggN4BgkqhkiG9w0BBWGg
ggNpBIIDZXsiaWV0Zi12b3VjaGVyOnZvdWNoZXIiOensiYXNzZXJ0aW9uIjoibG9nZ2VkIiwjY3Jl
YXRlZC1vbiI6IjIwMjAtMDItMjVUMTg6MDQ6NDkuMzAzLTA1OjAwIiwic2VyaWFsLW51bWJlciI6
IjAwLUQwLUU1LUYyLTAwLTAYIiwibm9uY2UiOiJhTWpndWVlVWQtMjJ3VmltajZ6MjdRIiwicGlu
bmVkbWVkbWVpbi1jZXJ0IjoiTU1JQi9EQ0NBWUtnQXJkFmSUVQNWliVWpBS0JnZ3Foa2pPUFFR
REFqQnRNUk13RUFZS0NaSW1pW1B5TEdRQkdSWUNZMkV4R1RBWEJnb0praWFKay9JclpBRVpGZ2x6
WVc1a1pXeHRZVzR4UERBNk1JnTlZCQU1NTTJadmRXNTBZV2x1TFhSbGMzUXVaWGhoY1hCc1pTNWpi
MjBnVlc1emRISjFibWNNUm05MWJuUmhhVzRnVW05dmRDQkRRVEFlRncweU1EQXlNa1V5TVRNeE5U
UmFGdzB5TWpBeU1qUXlNVE14TlRSYU1GTXhFakFRQmdvSmtYUprL0lZwKFFWkZnSmpZVEVaTUJj
R0NnbvNk12UOG14a0Fsa1dDWE5oYm1SbGJHMWhiakVpTUNBR0ExVUVBd3daWm05MWJuUmhhVzR0
ZEdWemRDNWx1R0Z0Y0d4bExtTnZiVEJaTUJNR0J5cUdTTTQ5QWdFR0NDcUdTTTQ5QXdfSEEWsUFC
SlpsVUhjMHVwL2wzZWpmOXZDQmIrbElub0VNRWdjN1JvK1haQ3RqQUkwQ0QxZkpmSlIvaE15eURt
SFd5WVW1ORmJSQ0g5Zn1hcmZremdYNHAWelRpenFqS2pBb01CWUdBMVVKs1LFFQi93UU1NQW9HQONz
R0FRVUZCd01jTUEOR0ExVWREd0VCL3dRRUF3SUhnREFLQmdncWhrak9QUVFEQWdOb0FEQmxBakJt
VDJCTVZVZ2VsZ2Y0M1IrNXlCS05SVGFibXl1QXZMdnh5ejBtRlZadlh4Ky8xUndPYWdtdkczYVht
UmtqLlg0Q01RQzhyTU5Cc0xvTnIxTDVURzU2ZndBZEK4aGlBV0c4UzhYQVI1azFDZ3gzWVVRQ1Nn
ZFNjRmNBZGYrK0J3N1l5K1U9In19oIIB4zCCAd8wggFkoAMCAQICBBuZXlQwCgYIKoZIzj0EAwIw
XTEPMA0GA1UEBhMGQ2FyYWRhMRAwDgYDVQQIDAdPbnRhcmlvMRIwEAYDVQQIDAlTYW5kZWxtYW4x
JDAiBgNVBAMMG2hpZ2h3YXktZGVzdC5leGFTcGx1LmNvbSBBDQTAeFw0xOTAyMTIyMjIyNDFAFw0y
MTAyMTEyMjIyNDFAFw0xOTAyMTIyMjIyNDFAFw0yMTAyMTEyMjIyNDFAFw0xOTAyMTIyMjIyNDFA
CwwJU2FuZGVsbWVwYXN5YwJAYDVQQDDDBloaWdod2F5LXRlc3QuZXhhbXBsZS5jb20gTUFTQTlBZMBMG
ByqGSM49AgEGCCqGSM49AwEHA0IABKoEFAneUEJE+Mn5GwcBpnRznB66bKmqzTCpoJJZ96AdRwFt
uTCVfoKouLTBX0idIhMLfJLM31lyuKy4CUTpp6WjEDAOMAwGA1UdEwEB/wQMAAwCgYIKoZIzj0EA
AwIDAQAwZGIXAL1V5ZsO+/xelSnjgbMVNaqTGKIEvkrYs1F9TW3r0dXBEDqyOXtXP8XMsKMO55lG
ugIXAPZ/RH23FPrRZ2rUEcNLrub7mphW+oUhlLxITPA/8ps/roggp675cv9b+XhozW9IyTGCATsw
ggE3AgEBMGUwXTEPMA0GA1UEBhMGQ2FyYWRhMRAwDgYDVQQIDAdPbnRhcmlvMRIwEAYDVQQIDAlT
YW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktZGVzdC5leGFTcGx1LmNvbSBBDQQIEG51fVDALBg1g
hkgBZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0yMDAy
MjUyMzA0NDlaMC8GCSqGSIb3DQEJBDEiBCCJQso4Z9msdaPk3bsDltTkVckX50DvOPuOR9Svi5M9
RDAKBggqhkiJOPQDQAgRHMEUCIQCKESXfM3iV8hpkqcxAKA1veArA6GFpN0jzys4E18uDgIgSRQi
9/MntuJhAM/tJCZBkfHBoAGX4PFAwwbs5LFZtAw=
<CODE ENDS>
```

The ASN1 decoding of the artifact:

file: examples/voucher_00-D0-E5-F2-00-02.b64

```
0:d=0  hl=4  l=1735  cons: SEQUENCE
4:d=1  hl=2  l= 9  prim: OBJECT               :pkcs7-signedData
15:d=1  hl=4  l=1720  cons: cont [ 0 ]
19:d=2  hl=4  l=1716  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= 888  cons: SEQUENCE
45:d=4  hl=2  l= 9  prim: OBJECT               :pkcs7-data
```

```

56:d=4  hl=4  l= 873 cons: cont [ 0 ]
60:d=5  hl=4  l= 869 prim: OCTET STRING      :{"ietf-voucher:voucher":
933:d=3  hl=4  l= 483 cons: cont [ 0 ]
937:d=4  hl=4  l= 479 cons: SEQUENCE
941:d=5  hl=4  l= 356 cons: SEQUENCE
945:d=6  hl=2  l=   3 cons: cont [ 0 ]
947:d=7  hl=2  l=   1 prim: INTEGER           :02
950:d=6  hl=2  l=   4 prim: INTEGER           :1B995F54
956:d=6  hl=2  l=  10 cons: SEQUENCE
958:d=7  hl=2  l=   8 prim: OBJECT             :ecdsa-with-SHA256
968:d=6  hl=2  l=  93 cons: SEQUENCE
970:d=7  hl=2  l=  15 cons: SET
972:d=8  hl=2  l=  13 cons: SEQUENCE
974:d=9  hl=2  l=   3 prim: OBJECT             :countryName
979:d=9  hl=2  l=   6 prim: PRINTABLESTRING    :Canada
987:d=7  hl=2  l=  16 cons: SET
989:d=8  hl=2  l=  14 cons: SEQUENCE
991:d=9  hl=2  l=   3 prim: OBJECT             :stateOrProvinceName
996:d=9  hl=2  l=   7 prim: UTF8STRING         :Ontario
1005:d=7  hl=2  l=  18 cons: SET
1007:d=8  hl=2  l=  16 cons: SEQUENCE
1009:d=9  hl=2  l=   3 prim: OBJECT             :organizationalUnitName
1014:d=9  hl=2  l=   9 prim: UTF8STRING         :Sandelman
1025:d=7  hl=2  l=  36 cons: SET
1027:d=8  hl=2  l=  34 cons: SEQUENCE
1029:d=9  hl=2  l=   3 prim: OBJECT             :commonName
1034:d=9  hl=2  l=  27 prim: UTF8STRING         :highway-test.example.com
1063:d=6  hl=2  l=  30 cons: SEQUENCE
1065:d=7  hl=2  l=  13 prim: UTCTIME           :190212222241Z
1080:d=7  hl=2  l=  13 prim: UTCTIME           :210211222241Z
1095:d=6  hl=2  l=  95 cons: SEQUENCE
1097:d=7  hl=2  l=  15 cons: SET
1099:d=8  hl=2  l=  13 cons: SEQUENCE
1101:d=9  hl=2  l=   3 prim: OBJECT             :countryName
1106:d=9  hl=2  l=   6 prim: PRINTABLESTRING    :Canada
1114:d=7  hl=2  l=  16 cons: SET
1116:d=8  hl=2  l=  14 cons: SEQUENCE
1118:d=9  hl=2  l=   3 prim: OBJECT             :stateOrProvinceName
1123:d=9  hl=2  l=   7 prim: UTF8STRING         :Ontario
1132:d=7  hl=2  l=  18 cons: SET
1134:d=8  hl=2  l=  16 cons: SEQUENCE
1136:d=9  hl=2  l=   3 prim: OBJECT             :organizationalUnitName
1141:d=9  hl=2  l=   9 prim: UTF8STRING         :Sandelman
1152:d=7  hl=2  l=  38 cons: SET
1154:d=8  hl=2  l=  36 cons: SEQUENCE
1156:d=9  hl=2  l=   3 prim: OBJECT             :commonName
1161:d=9  hl=2  l=  29 prim: UTF8STRING         :highway-test.example.com
1192:d=6  hl=2  l=  89 cons: SEQUENCE

```

```
1194:d=7  hl=2 l= 19 cons: SEQUENCE
1196:d=8  hl=2 l=  7 prim: OBJECT           :id-ecPublicKey
1205:d=8  hl=2 l=  8 prim: OBJECT           :prime256v1
1215:d=7  hl=2 l= 66 prim: BIT STRING
1283:d=6  hl=2 l= 16 cons: cont [ 3 ]
1285:d=7  hl=2 l= 14 cons: SEQUENCE
1287:d=8  hl=2 l= 12 cons: SEQUENCE
1289:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Basic Constraints
1294:d=9  hl=2 l=  1 prim: BOOLEAN          :255
1297:d=9  hl=2 l=  2 prim: OCTET STRING     [HEX DUMP]:3000
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= 105 prim: BIT STRING
1420:d=3  hl=4 l= 315 cons: SET
1424:d=4  hl=4 l= 311 cons: SEQUENCE
1428:d=5  hl=2 l=  1 prim: INTEGER           :01
1431:d=5  hl=2 l= 101 cons: SEQUENCE
1433:d=6  hl=2 l=  93 cons: SEQUENCE
1435:d=7  hl=2 l=  15 cons: SET
1437:d=8  hl=2 l=  13 cons: SEQUENCE
1439:d=9  hl=2 l=  3 prim: OBJECT           :countryName
1444:d=9  hl=2 l=  6 prim: PRINTABLESTRING  :Canada
1452:d=7  hl=2 l= 16 cons: SET
1454:d=8  hl=2 l= 14 cons: SEQUENCE
1456:d=9  hl=2 l=  3 prim: OBJECT           :stateOrProvinceName
1461:d=9  hl=2 l=  7 prim: UTF8STRING       :Ontario
1470:d=7  hl=2 l= 18 cons: SET
1472:d=8  hl=2 l= 16 cons: SEQUENCE
1474:d=9  hl=2 l=  3 prim: OBJECT           :organizationalUnitName
1479:d=9  hl=2 l=  9 prim: UTF8STRING       :Sandelman
1490:d=7  hl=2 l= 36 cons: SET
1492:d=8  hl=2 l= 34 cons: SEQUENCE
1494:d=9  hl=2 l=  3 prim: OBJECT           :commonName
1499:d=9  hl=2 l= 27 prim: UTF8STRING       :highway-test.example.com
1528:d=6  hl=2 l=  4 prim: INTEGER           :1B995F54
1534:d=5  hl=2 l= 11 cons: SEQUENCE
1536:d=6  hl=2 l=  9 prim: OBJECT           :sha256
1547:d=5  hl=2 l= 105 cons: cont [ 0 ]
1549:d=6  hl=2 l= 24 cons: SEQUENCE
1551:d=7  hl=2 l=  9 prim: OBJECT           :contentType
1562:d=7  hl=2 l= 11 cons: SET
1564:d=8  hl=2 l=  9 prim: OBJECT           :pkcs7-data
1575:d=6  hl=2 l= 28 cons: SEQUENCE
1577:d=7  hl=2 l=  9 prim: OBJECT           :signingTime
1588:d=7  hl=2 l= 15 cons: SET
1590:d=8  hl=2 l= 13 prim: UTCTIME          :200225230449Z
1605:d=6  hl=2 l= 47 cons: SEQUENCE
1607:d=7  hl=2 l=  9 prim: OBJECT           :messageDigest
```

```
1618:d=7  hl=2 l= 34 cons: SET
1620:d=8  hl=2 l= 32 prim: OCTET STRING      [HEX DUMP]:8942CA3867D9AC
1654:d=5  hl=2 l= 10 cons: SEQUENCE
1656:d=6  hl=2 l=  8 prim: OBJECT            :ecdsa-with-SHA256
1666:d=5  hl=2 l= 71 prim: OCTET STRING      [HEX DUMP]:30450221008A11
```

Appendix D. Additional References

RFC EDITOR Please remove this section before publication. It exists just to include references to the things in the YANG descriptions which are not otherwise referenced in the text so that xml2rfc will not complain.

[ITU.X690.1994]

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/>

Toerless Eckert
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara, CA 95050
United States of America

Email: tte+ietf@cs.fau.de

Michael H. Behringer

Email: Michael.H.Behringer@gmail.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

anima Working Group
Internet-Draft
Updates: 8366, 8995 (if approved)
Intended status: Standards Track
Expires: 9 October 2022

M. Richardson
Sandelman Software Works
P. van der Stok
vanderstok consultancy
P. Kampanakis
Cisco Systems
E. Dijk
IoTconsultancy.nl
7 April 2022

Constrained Bootstrapping Remote Secure Key Infrastructure (BRSKI)
draft-ietf-anima-constrained-voucher-17

Abstract

This document defines the Constrained Bootstrapping Remote Secure Key Infrastructure (Constrained BRSKI) protocol, which provides a solution for secure zero-touch bootstrapping of resource-constrained (IoT) devices into the network of a domain owner. This protocol is designed for constrained networks, which may have limited data throughput or may experience frequent packet loss. Constrained BRSKI is a variant of the BRSKI protocol, which uses an artifact signed by the device manufacturer called the "voucher" which enables a new device and the owner's network to mutually authenticate. While the BRSKI voucher is typically encoded in JSON, Constrained BRSKI defines a compact CBOR-encoded voucher. The BRSKI voucher is extended with new data types that allow for smaller voucher sizes. The Enrollment over Secure Transport (EST) protocol, used in BRSKI, is replaced with EST-over-CoAPS; and HTTPS used in BRSKI is replaced with CoAPS.

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 9 October 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
2. Terminology	5
3. Requirements Language	6
4. Overview of Protocol	6
5. Updates to RFC8366 and RFC8995	7
6. BRSKI-EST Protocol	8
6.1. Registrar and the Server Name Indicator (SNI)	8
6.2. TLS Client Certificates: IDevID authentication	9
6.3. Discovery, URIs and Content Formats	10
6.3.1. RFC8995 Telemetry Returns	12
6.4. Join Proxy options	13
6.5. Extensions to BRSKI	13
6.5.1. Discovery	13
6.5.2. CoAP responses	13
6.6. Extensions to EST-coaps	14
6.6.1. Pledge Extensions	15
6.6.2. EST-client Extensions	16
6.6.3. Registrar Extensions	18
6.7. DTLS handshake fragmentation Considerations	19
7. BRSKI-MASA Protocol	19
7.1. Protocol and Formats	19
7.2. Registrar Voucher Request	20
7.3. MASA and the Server Name Indicator (SNI)	20
7.3.1. Registrar Certificate Requirement	21
8. Pinning in Voucher Artifacts	21
8.1. Registrar Identity Selection and Encoding	21
8.2. MASA Pinning Policy	23
8.3. Pinning of Raw Public Keys	24
8.4. Considerations for use of IDevID-Issuer	25
9. Artifacts	26
9.1. Voucher Request artifact	26
9.1.1. Tree Diagram	26

9.1.2.	SID values	27
9.1.3.	YANG Module	28
9.1.4.	Example voucher request artifact	32
9.2.	Voucher artifact	32
9.2.1.	Tree Diagram	32
9.2.2.	SID values	33
9.2.3.	YANG Module	33
9.2.4.	Example voucher artifacts	36
9.3.	Signing voucher and voucher-request artifacts with COSE	37
10.	Deployment-specific Discovery Considerations	38
10.1.	6TSCH Deployments	39
10.2.	Generic networks using GRASP	39
10.3.	Generic networks using mDNS	39
10.4.	Thread networks using Mesh Link Establishment (MLE)	39
10.5.	Non-mesh networks using CoAP Discovery	40
11.	Design Considerations	40
12.	Raw Public Key Use Considerations	40
12.1.	The Registrar Trust Anchor	40
12.2.	The Pledge Voucher Request	41
12.3.	The Voucher Response	41
13.	Use of constrained vouchers with HTTPS	41
14.	Security Considerations	42
14.1.	Duplicate serial-numbers	42
14.2.	IDevID security in Pledge	43
14.3.	Security of CoAP and UDP protocols	44
14.4.	Registrar Certificate may be self-signed	45
14.5.	Use of RPK alternatives to proximity-registrar-cert	45
14.6.	MASA support of CoAPS	46
15.	IANA Considerations	46
15.1.	Resource Type Registry	46
15.2.	The IETF XML Registry	47
15.3.	The YANG Module Names Registry	47
15.4.	The RFC SID range assignment sub-registry	47
15.5.	Media Types Registry	48
15.5.1.	application/voucher-cose+cbor	48
15.6.	CoAP Content-Format Registry	48
15.7.	Update to BRSKI Parameters Registry	49
16.	Acknowledgements	49
17.	Changelog	50
18.	References	50
18.1.	Normative References	50
18.2.	Informative References	54
Appendix A.	Library Support for BRSKI	56
A.1.	OpenSSL	57
A.2.	mbedtls	58
Appendix B.	Constrained BRSKI-EST Message Examples	59
B.1.	enrollstatus	59

B.2. voucher_status	60
Appendix C. COSE-signed Voucher (Request) Examples	61
C.1. Pledge, Registrar and MASA Keys	61
C.1.1. Pledge IDevID private key	61
C.1.2. Registrar private key	61
C.1.3. MASA private key	62
C.2. Pledge, Registrar and MASA Certificates	62
C.2.1. Pledge IDevID Certificate	62
C.2.2. Registrar Certificate	64
C.2.3. MASA Certificate	66
C.3. COSE-signed Pledge Voucher Request (PVR)	68
C.4. COSE-signed Registrar Voucher Request (RVR)	70
C.5. COSE-signed Voucher from MASA	72
Appendix D. Generating Certificates with OpenSSL	74
Appendix E. Pledge Device Class Profiles	77
E.1. Minimal Pledge	78
E.2. Typical Pledge	78
E.3. Full-featured Pledge	78
E.4. Comparison Chart of Pledge Classes	78
Contributors	79
Authors' Addresses	80

1. Introduction

Secure enrollment of new nodes into constrained networks with constrained nodes presents unique challenges. As explained in [RFC7228], the networks are challenged and the nodes are constrained by energy, memory space, and code size.

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol described in [RFC8995] provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices. In it, new devices, such as IoT devices, are called "pledges", and equipped with a factory-installed Initial Device Identifier (IDevID) (see [ieee802-1AR]), are enrolled into a network.

The BRSKI solution described in [RFC8995] was designed to be modular, and this document describes a version scaled to the constraints of IoT deployments.

Therefore, this document defines a constrained version of the voucher artifact (described in [RFC8366]), along with a constrained version of BRSKI. This constrained-BRSKI protocol makes use of the constrained CoAP-based version of EST (EST-coaps from [I-D.ietf-ace-coap-est]) rather than the EST over HTTPS [RFC7030]. Constrained-BRSKI is itself scalable to multiple resource levels through the definition of optional functions. Appendix E illustrates this.

In BRSKI, the [RFC8366] voucher is by default serialized to JSON with a signature in CMS [RFC5652]. This document defines a new voucher serialization to CBOR [RFC8949] with a signature in COSE [I-D.ietf-cose-rfc8152bis-struct].

This COSE-signed CBOR-encoded voucher is transported using both secured CoAP and HTTPS. The CoAP connection (between Pledge and Registrar) is to be protected by either OSCORE+EDHOC [I-D.ietf-lake-edhoc] or DTLS (CoAPS). The HTTP connection (between Registrar and MASA) is to be protected using TLS (HTTPS).

This document specifies a constrained voucher-request artifact based on Section 3 of [RFC8995], and voucher(-request) transport over CoAP based on Section 3 of [RFC8995] and on [I-D.ietf-ace-coap-est].

The CBOR definitions for the constrained voucher format are defined using the mechanism described 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 a list of SID keys is still in its infancy, the table of SID values presented here MUST be considered normative rather than the output of the tool specified in [I-D.ietf-core-sid].

2. Terminology

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

The following terms from [RFC8995] are used identically as in that document: Domain CA, enrollment, IDevID, Join Proxy, LDevID, manufacturer, nonced, nonceless, PKIX.

The term Pledge Voucher Request, or acronym PVR, is introduced to refer to the voucher request between the pledge and the Registrar.

The term Registrar Voucher Request, or acronym RVR, is introduced to refer to the voucher request between the Registrar and the MASA.

In code examples, the string "<CODE BEGINS>" denotes the start of a code example and "<CODE ENDS>" the end of the code example. Four dots ("....") in a CBOR diagnostic notation byte string denotes a further sequence of bytes that is not shown for brevity.

3. Requirements Language

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

4. Overview of Protocol

[RFC8366] provides for vouchers that assert proximity, authenticate the Registrar, and can offer varying levels of anti-replay protection.

The proximity proof provided for in [RFC8366], is an assertion that the Pledge and the Registrar are believed to be close together, from a network topology point of view. Like in [RFC8995], proximity is shown by making TLS connections between the Pledge and Registrar using IPv6 Link-Local addresses.

The TLS connection is used to make a Voucher Request. This request is verified by an agent of the Pledge's manufacturer, which then issues a voucher. The voucher provides an authorization statement from the manufacturer indicating that the Registrar is the intended owner of the device. The voucher refers to the Registrar through pinning of the Registrar's identity.

This document does not make any extensions to the semantic meaning of vouchers, only the encoding has been changed to optimize for constrained devices and networks. The two main parts of the BRSKI protocol are named separately in this document: BRSKI-EST for the protocol between Pledge and Registrar, and BRSKI-MASA for the protocol between the Registrar and the MASA.

Time-based vouchers are supported in this definition, but given that constrained devices are extremely unlikely to have accurate time, their use will be uncommon. Most Pledges using constrained vouchers will be online during enrollment and will use live nonces to provide anti-replay protection rather than expiry times.

[RFC8366] defines the voucher artifact, while the Voucher Request artifact was defined in [RFC8995]. This document defines both a constrained voucher and a constrained voucher-request. They are presented in the order "voucher-request", followed by a "voucher" response as this is the order that they occur in the protocol.

The constrained voucher request MUST be signed by the Pledge. It signs using the private key associated with its IDevID X.509 certificate, or if an IDevID is not available, then the private key associated with its manufacturer-installed raw public key (RPK). Section 12 provides additional details on PKIX-less operations.

The constrained voucher MUST be signed by the MASA.

For the constrained voucher request this document defines two distinct methods for the Pledge to identify the Registrar: using either the Registrar's X.509 certificate, or using a raw public key (RPK) of the Registrar.

For the constrained voucher both methods are supported to indicate (pin) a trusted domain identity: using either a pinned domain X.509 certificate, or a pinned raw public key (RPK).

The BRSKI architectures mandates that the MASA be aware of the capabilities of the pledge. This is not a drawback as the pledges are constructed by a manufacturer which also arranges for the MASA to be aware of the inventory of devices.

The MASA therefore knows if the pledge supports PKIX operations, PKIX format certificates, or if the pledge is limited to Raw Public Keys (RPK). Based upon this, the MASA can select which attributes to use in the voucher for certain operations, like the pinning of the Registrar identity. This is described in more detail in Section 9.2.3, Section 8 and Section 8.3 (for RPK specifically).

5. Updates to RFC8366 and RFC8995

This section details the ways in which this document updates other RFCs. The terminology for Updates is taken from [I-D.kuehlewind-update-tag].

This document Updates [RFC8366]. It Extends [RFC8366] by creating a new serialization format, and creates a mechanism to pin Raw Public Key (RPK).

This document Updates [RFC8995]. It Amends [RFC8995]

- * by clarifying how pinning is done,
- * adopts clearer explanation of the TLS Server Name Indicator (SNI), see Section 6.1 and Section 7.3
- * clarifies when new trust anchors should be retrieved (Section 6.6.1),

- * clarified what kinds of Extended Key Usage attributes are appropriate for each certificate (Section 7.3.1)

It Extends [RFC8995] as follows: * defines the CoAP version of the BRSKI protocol

- * makes some messages optional if the results can be inferred from other validations (Section 6.6),
- * provides the option to return trust anchors in a simpler format (Section 6.6.3)
- * extends the BRSKI-MASA protocol to carry the new voucher-cose+cbor format.

6. BRSKI-EST Protocol

This section describes the constrained BRSKI extensions to EST-coaps [I-D.ietf-ace-coap-est] to transport the voucher between Registrar and Pledge (optionally via a Join Proxy) over CoAP. The extensions are targeting low-resource networks with small packets.

The constrained BRSKI-EST protocol described in this section is between the Pledge and the Registrar only.

6.1. Registrar and the Server Name Indicator (SNI)

A DTLS connection is established between the Pledge and the Registrar, similar to the TLS connection described in Section 5.1 of [RFC8995]. This may occur via a Join Proxy as described in Section 6.4. Regardless of the Join Proxy mechanism, the DTLS connection should operate identically.

The SNI issue described below affects [RFC8995] as well, and is reported in errata: <https://www.rfc-editor.org/errata/eid6648>

As the Registrar is discovered by IP address, and typically connected via a Join Proxy, the name of the Registrar is not known to the Pledge. The Pledge will not know what the hostname for the Registrar is, so it cannot do RFC6125 DNS-ID validation on the Registrar's certificate. Instead, it must do validation using the RFC8366 voucher.

As the Pledge does not know the name of the Registrar, the Pledge cannot put any reasonable value into the [RFC6066] Server Name Indicator (SNI). Therefore the Pledge SHOULD omit the SNI extension as per Section 9.2 of [RFC8446].

In some cases, particularly while testing BRSKI, a Pledge may be given the hostname of a particular Registrar to connect to directly. Such a bypass of the discovery process may result in the Pledge taking a different code branch to establish a DTLS connection, and may result in the SNI being inserted by a library. The Registrar MUST ignore any SNI seen.

A primary motivation for making the SNI ubiquitous in the public web is because it allows for multi-tenant hosting of HTTPS sites on a single (scarce) IPv4 address. This consideration does not apply to the server function in the Registrar because:

- * it uses DTLS and CoAP, not HTTPS
- * it typically uses IPv6, often [RFC4193] Unique Local Address, which are plentiful
- * the server port number is typically discovered, so multiple tenants can be accommodated via unique port numbers.

As per Section 3.6.1 of [RFC7030], the Registrar certificate MUST have the Extended Key Usage (EKU) id-kp-cmcRA. This certificate is also used as a TLS Server Certificate, so it MUST also have the EKU id-kp-serverAuth.

6.2. TLS Client Certificates: IDevID authentication

As described in Section 5.1 of [RFC8995], the Pledge makes a connection to the Registrar using a TLS Client Certificate for authentication.

Subsequently the Pledge will send a Pledge Voucher Request (PVR).

As explained below in Section 8.1, the "x5bag" element may be used in the RVR to communicate identity of the Registrar to MASA. The Pledge SHOULD NOT use the x5bag attribute in this way in the PVR. A Registrar that processes a PVR with an x5bag attribute MUST ignore it, and MUST use only the TLS Client Certificate extension for authentication of the Pledge.

A situation where the Pledge MAY use the x5bag is for communication of certificate chains to the MASA. This would arise in some vendor-specific situations involving outsourcing of MASA functionality, or rekeying of the IDevID certification authority.

6.3. Discovery, URIs and Content Formats

To keep the protocol messages small the EST-coaps and constrained-BRSKI URIs are shorter than the respective EST and BRSKI URIs.

The EST-coaps server URIs differ from the EST URIs by replacing the scheme https by coaps and by specifying shorter resource path names. Below are some examples; the first two using a discovered short path name and the last one using the well-known URI of EST which requires no discovery.

```
coaps://server.example.com/est/<short-name>
coaps://server.example.com/e/<short-name>
coaps://server.example.com/.well-known/est/<short-name>
```

Similarly the constrained BRSKI server URIs differ from the BRSKI URIs by replacing the scheme https by coaps and by specifying shorter resource path names. Below are some examples; the first two using a discovered short path name and the last one using the well-known URI prefix which requires no discovery. This is the same `"/.well-known/brski"` prefix as defined in Section 5 of [RFC8995].

```
coaps://server.example.com/brski/<short-name>
coaps://server.example.com/b/<short-name>
coaps://server.example.com/.well-known/brski/<short-name>
```

Figure 5 in Section 3.2.2 of [RFC7030] enumerates the operations supported by EST, for which Table 1 in Section 5.1 of [I-D.ietf-ace-coap-est] enumerates the corresponding EST-coaps short path names. Similarly, Table 1 below provides the mapping from the supported BRSKI extension URI paths to the constrained-BRSKI URI paths.

=====	=====
BRSKI resource constrained-BRSKI resource	
=====	=====
/requestvoucher /rv	
-----	-----
/voucher_status /vs	
-----	-----
/enrollstatus /es	
-----	-----

Table 1: BRSKI URI paths mapping to
Constrained BRSKI URI paths

Note that /requestvoucher indicated above occurs between the Pledge and Registrar (in scope of the BRSKI-EST protocol), but it also occurs between Registrar and MASA. However, as described in Section 6, this section and above table addresses only the BRSKI-EST protocol.

Pledges that wish to discover the available BRSKI bootstrap options/formats, or reduce the size of the CoAP headers by eliminating the "/.well-known/brski" path, can do a discovery operation using [RFC6690] Section 4 by sending a discovery query to the Registrar.

For example, if the Registrar supports a short BRSKI URL (/b) and supports the voucher format "application/voucher-cose+cbor" (TBD3), and status reporting in both CBOR and JSON formats:

```
REQ: GET /.well-known/core?rt=brski*
```

```
RES: 2.05 Content
```

```
Content-Format: 40
```

```
Payload:
```

```
</b>;rt=brski,  
</b/rv>;rt=brski.rv;ct=TBD3,  
</b/vs>;rt=brski.vs;ct="50 60",  
</b/es>;rt=brski.es;ct="50 60"
```

The Registrar is under no obligation to provide shorter URLs, and MAY respond to this query with only the "/.well-known/brski/<short-name>" resources for the short names as defined in Table 1.

Registrars that have implemented shorter URLs MUST also respond in equivalent ways to the corresponding "/.well-known/brski/<short-name>" URLs, and MUST NOT distinguish between them. In particular, a Pledge MAY use the longer and shorter URLs in any combination.

When responding to a discovery request for BRSKI resources, the server MAY in addition return the full resource paths and the content types which are supported by these resources as shown in above example. This is useful when multiple content types are specified for a particular resource on the server. The server responds with only the root path for the BRSKI resources (rt=brski, resource /b in above example) and no others in case the client queries for only rt=brski type resources. (So, a query for rt=brski, without the wildcard character.)

Without discovery, a longer well-known URL can only be used, such as:

```
REQ: GET /.well-known/brski/rv
```

while with discovery of shorter URLs, a request such as:

```
REQ: GET /b/rv
```

is possible.

The return of multiple content-types in the "ct" attribute allows the Pledge to choose the most appropriate one. Note that Content-Format TBD3 ("application/voucher-cose+cbor") is defined in this document.

Content-Format TBD3 MUST be supported by the Registrar for the /rv resource. If the "ct" attribute is not indicated for the /rv resource in the link format description, this implies that at least TBD3 is supported.

Note that this specification allows for voucher-cose+cbor format requests and vouchers to be transmitted over HTTPS, as well as for voucher-cms+json and other formats yet to be defined over CoAP. The burden for this flexibility is placed upon the Registrar. A Pledge on constrained hardware is expected to support a single format only.

The Pledge and MASA need to support one or more formats (at least TBD3) for the voucher and for the voucher request. The MASA needs to support all formats that the Pledge supports.

Section 10 details how the Pledge discovers the Registrar and Join Proxy in different deployment scenarios.

6.3.1. RFC8995 Telemetry Returns

[RFC8995] defines two telemetry returns from the Pledge which are sent to the Registrar. These are the BRSKI Status Telemetry [RFC8995], Section 5.7 and the Enrollment Status Telemetry [RFC8995], Section 5.9.4. These are two POST operations made the by Pledge at two key steps in the process.

[RFC8995] defines the content of these POST operations in CDDL, which are serialized as JSON. This document extends the list of acceptable formats to CBOR as well as JSON, using the rules from [RFC8610].

The existing JSON format is described as CoAP Content-Format 50 ("application/json"), and it MAY be supported. The new CBOR format described as CoAP Content-Format 60 ("application/cbor"), MUST be supported by the Registrar for both the /vs and /es resources.

6.4. Join Proxy options

[I-D.ietf-anima-constrained-join-proxy] specifies a constrained Join Proxy that is optionally placed between Pledge and Registrar. This includes methods for discovery of the Join Proxy by the Pledge and discovery of the Registrar by the Join Proxy.

6.5. Extensions to BRSKI

6.5.1. Discovery

The Pledge discovers an IP address and port number that connects to the Registrar (possibly via a Join Proxy), and it establishes a DTLS connection.

No further discovery of hosts or port numbers is required, but a pledge that can do more than one kind of enrollment (future work offers protocols other than [I-D.ietf-ace-coap-est]), then a pledge may need to use CoAP Discovery to determine what other protocols are available.

A Pledge that only supports the EST-coaps enrollment method SHOULD NOT use discovery for BRSKI resources. It is more efficient to just try the supported enrollment method via the well-known BRSKI/EST-coaps resources. This also avoids the Pledge doing any CoRE Link Format parsing, which is specified in [I-D.ietf-ace-coap-est], Section 4.1.

The Registrar MUST support all of the EST resources at their default ".well-known" locations (on the specified port) as well as any server-specific shorter form that might also be supported.

However, when discovery is being done by the Pledge, it is possible for the Registrar to return references to resources which are on different port numbers. The Registrar SHOULD NOT use different ports numbers by default, because a Pledge that is connected via a Join Proxy can only access a single UDP port. A Registrar configured to never use Join Proxies MAY be configured to use multiple port numbers. Therefore a Registrar MUST host all discoverable BRSKI resources on the same (UDP) server port that the Pledge's DTLS connection is using. Using the same UDP server port for all resources allows the Pledge to continue via the same DTLS connection which is more efficient.

6.5.2. CoAP responses

[RFC8995], Section 5 defines a number of HTTP response codes that the Registrar is to return when certain conditions occur.

The 401, 403, 404, 406 and 415 response codes map directly to CoAP codes 4.01, 4.03, 4.04, 4.06 and 4.15.

The 202 Retry process which occurs in the voucher request, is to be handled in the same way as Section 5.7 of [I-D.ietf-ace-coap-est] process for Delayed Responses.

6.6. Extensions to EST-coaps

This document extends [I-D.ietf-ace-coap-est], and it inherits the functions described in that document: specifically, the mandatory Simple (Re-)Enrollment (/sen and /sren) and Certification Authority certificates request (/crtcs). Support for CSR Attributes Request (/att) and server-side key generation (/skg, /skc) remains optional for the EST server.

Collecting the resource definitions from both [RFC8995], [RFC7030], and [I-D.ietf-ace-coap-est] results in the following shorter forms of URI paths for the commonly used resources:

=====	=====	=====
BRSKI + EST	Constrained-BRSKI + EST	Well-known URI namespace
=====	=====	=====
/requestvoucher	/rv	brski
/voucher_status	/vs	brski
/csrattrs	/att	est
/simpleenroll	/sen	est
/cacerts	/crtcs	est
/enrollstatus	/es	brski
/simplereenroll	/sren	est
=====	=====	=====

Table 2: BRSKI/EST URI paths mapping to Constrained BRSKI/EST short URI paths

6.6.1. Pledge Extensions

This section defines extensions to the BRSKI Pledge, which are applicable during the BRSKI bootstrap procedure. A Pledge which only supports the EST-coaps enrollment method, SHOULD NOT use discovery for EST-coaps resources, because it is more efficient to enroll (e.g. /sen) via the well-known EST resource on the current DTLS connection. This avoids an additional round-trip of packets and avoids the Pledge having to unnecessarily implement CoRE Link Format parsing.

A constrained Pledge SHOULD NOT perform the optional EST "CSR attributes request" (/att) to minimize network traffic. The Pledge selects which attributes to include in the CSR.

One or more Subject Distinguished Name fields MUST be included. If the Pledge has no specific information on what attributes/fields are desired in the CSR, it MUST use the Subject Distinguished Name fields from its LDevID unmodified. The Pledge can receive such information via the voucher (encoded in a vendor-specific way) or via some other, out-of-band means.

A constrained Pledge MAY use the following optimized EST-coaps procedure to minimize network traffic.

1. if the voucher, that validates the current Registrar, contains a single pinned domain CA certificate, the Pledge provisionally considers this certificate as the EST trust anchor, as if it were the result of "CA certificates request" (/crts) to the Registrar.
2. Using this CA certificate as trust anchor it proceeds with EST simple enrollment (/sen) to obtain its provisionally trusted LDevID certificate.
3. If the Pledge validates that the trust anchor CA was used to sign its LDevID certificate, the Pledge accepts the pinned domain CA certificate as the legitimate trust anchor CA for the Registrar's domain and accepts the associated LDevID certificate.
4. If the trust anchor CA was NOT used to sign its LDevID certificate, the Pledge MUST perform an actual "CA certificates request" (/crts) to the EST server to obtain the EST CA trust anchor(s) since these can differ from the (temporary) pinned domain CA.

5. When doing this /crtts request, the Pledge MAY use a CoAP Accept Option with value TBD287 ("application/pkix-cert") to limit the number of returned EST CA trust anchors to only one. A constrained Pledge MAY support only this format in a /crtts response, per Section 5.3 of [I-D.ietf-ace-coap-est].
6. If the Pledge cannot obtain the single CA certificate or the finally validated CA certificate cannot be chained to the LDevID certificate, then the Pledge MUST abort the enrollment process and report the error using the enrollment status telemetry (/es).

Note that even though the Pledge may avoid performing any /crtts request using the above EST-coaps procedure during bootstrap, it SHOULD support retrieval of the trust anchor CA periodically as detailed in the next section.

6.6.2. EST-client Extensions

This section defines extensions to EST-coaps clients, used after the BRSKI bootstrap procedure is completed. (Note that such client is not called "Pledge" in this section, since it is already enrolled into the domain.) A constrained EST-coaps client MAY support only the Content-Format TBD287 ("application/pkix-cert") in a /crtts response, per Section 5.3 of [I-D.ietf-ace-coap-est]. In this case, it can only store one trust anchor of the domain.

An EST-coaps client that has an idea of the current time (internally, or via NTP) SHOULD consider the validity time of the trust anchor CA, and MAY begin requesting a new trust anchor CA using the /crtts request when the CA has 50% of its validity time (notAfter - notBefore) left. A client without access to the current time cannot decide if the trust anchor CA has expired, and SHOULD poll periodically for a new trust anchor using the /crtts request at an interval of approximately 1 month. An EST-coaps server SHOULD include the CoAP ETag Option in every response to a /crtts request, to enable clients to perform low-overhead validation whether their trust anchor CA is still valid. The EST-coaps client SHOULD store the ETag resulting from a /crtts response in memory and SHOULD use this value in an ETag Option in its next GET /crtts request.

The above-mentioned limitation that an EST-coaps client may support only one trust anchor CA is not an issue in case the domain trust anchor remains stable. However, special consideration is needed for cases where the domain trust anchor can change over time. Such a change may happen due to relocation of the client device to a new domain, or due to key update of the trust anchor as described in [RFC4210], Section 4.4.

From the client's viewpoint, a trust anchor change typically happens during EST re-enrollment: a change of domain CA requires all devices operating under the old domain CA to acquire a new LDevID issued by the new domain CA. A client's re-enrollment may be triggered by various events, such as an instruction to re-enroll sent by a domain entity, or an imminent expiry of its LDevID certificate. How the re-enrollment is explicitly triggered on the client by a domain entity, such as a commissioner or a Registrar, is out of scope of this specification.

The mechanism described in [RFC4210], Section 4.4 for Root CA key update requires four certificates: OldWithOld, OldWithNew, NewWithOld, and NewWithNew. The OldWithOld certificate is already stored in the EST client's trust store. The NewWithNew certificate will be distributed as the single certificate in a /crls response, during EST re-enrollment. Since the EST client can only accept a single certificate in a /crls response it implies that the EST client cannot obtain the certificates OldWithNew and NewWithOld in this way, to perform the complete verification of the new domain CA. Instead, the client only verifies the EST server (Registrar) using its old domain CA certificate in its trust store as detailed below, and based on this trust in the active and valid DTLS connection it automatically trusts the new (NewWithNew) domain CA certificate that the EST server provides in the /crls response.

In this manner, even during rollover of trust anchors, it is possible to have only a single trust anchor provided in a /crls response.

During the period of the certificate renewal, it is not possible to create new communication channels between devices with NewCA certificates devices with OldCA certificates. One option is that devices should avoid restarting existing DTLS or OSCORE connections during this interval that new certificates are being deployed. The recommended period for certificate renewal is 24 hours. For re-enrollment, the constrained EST-coaps client MUST support the following EST-coaps procedure, where optional re-enrollment to a new domain is under control of the Registrar:

1. The client connects with DTLS to the Registrar, and authenticates with its present domain certificate (LDevID certificate) as usual. The Registrar authenticates itself with its domain certificate that is trusted by the client, i.e. it chains to the single trust anchor that the client has stored. This is the "old" trust anchor, the one that will be eventually replaced in case the Registrar decides to re-enroll the client into a new domain.

2. The client performs the simple re-enrollment request (/sren) and upon success it obtains a new LDevID.
3. The client verifies the new LDevID against its (single) existing domain trust anchor. If it chains successfully, this means the trust anchor did not change and the client MAY skip retrieving the current CA certificate using the "CA certificates request" (/crts). If it does not chain successfully, this means the trust anchor was changed/updated and the client then MUST retrieve the new domain trust anchor using the "CA certificates request" (/crts).
4. If the client retrieved a new trust anchor in step 3, then it MUST verify that the new trust anchor chains with the new LDevID certificate it obtained in step 2. If it chains successfully, the client stores both, accepts the new LDevID certificate and stops using its prior LDevID certificate. If it does not chain successfully, the client MUST NOT update its LDevID certificate, it MUST NOT update its (single) domain trust anchor, and the client MUST abort the enrollment process and report the error to the Registrar using enrollment status telemetry (/es).

Note that even though the EST-coaps client may skip the /crts request in step 3, it SHOULD support retrieval of the trust anchor CA periodically as detailed earlier in this section.

6.6.3. Registrar Extensions

A Registrar SHOULD host any discoverable EST-coaps resources on the same (UDP) server port that the Pledge's DTLS initial connection is using. This avoids the overhead of the Pledge reconnecting using DTLS, when it performs EST enrollment after the BRSKI voucher request.

The Content-Format 50 (application/json) MUST be supported and 60 (application/cbor) MUST be supported by the Registrar for the /vs and /es resources.

Content-Format TBD3 MUST be supported by the Registrar for the /rv resource.

When a Registrar receives a "CA certificates request" (/crts) request with a CoAP Accept Option with value TBD287 ("application/pkix-cert") it SHOULD return only the single CA certificate that is the envisioned or actual authority for the current, authenticated Pledge making the request.

If the Pledge included in its request an Accept Option for only the TBD287 ("application/pkix-cert") Content Format, but the domain has been configured to operate with multiple CA trust anchors only, then the Registrar returns a 4.06 Not Acceptable error to signal that the Pledge needs to use the Content Format 281 ("application/pkcs7-mime; smime-type=certs-only") to retrieve all the certificates.

If the current authenticated client is an EST-coaps client that was already enrolled in the domain, and the Registrar is configured to assign this client to a new domain CA trust anchor during the next EST re-enrollment procedure, then the Registrar MUST respond with the new domain CA certificate in case the client performs the "CA Certificates request" (/crt) with an Accept Option for TBD287 only. This signals the client that a new domain is assigned to it. The client follows the procedure as defined in Section 6.6.2.

6.7. DTLS handshake fragmentation Considerations

DTLS includes a mechanism to fragment the handshake messages. This is described in Section 4.4 of [I-D.ietf-tls-dtls13]. The protocol described in this document will often be used with a Join Proxy described in [I-D.ietf-anima-constrained-join-proxy]. The Join Proxy will need some overhead, while the maximum packet sized guaranteed on 802.15.4 networks is 1280 bytes. It is RECOMMENDED that a PMTU of 1024 bytes be assumed for the DTLS handshake. It is unlikely that any Packet Too Big indications [RFC4443] will be relayed by the Join Proxy.

During the operation of the constrained BRSKI-EST protocol, the CoAP Blockwise transfer mechanism will be used when message sizes exceed the PMTU. A Pledge/EST-client on a constrained network MUST use the (D)TLS maximum fragment length extension ("max_fragment_length") defined in Section 4 of [RFC6066] with the maximum fragment length set to a value of either 2^9 or 2^{10} .

7. BRSKI-MASA Protocol

This section describes extensions to and clarifications of the BRSKI-MASA protocol between Registrar and MASA.

7.1. Protocol and Formats

Section 5.4 of [RFC8995] describes a connection between the Registrar and the MASA as being a normal TLS connection using HTTPS. This document does not change that. The Registrar MUST use the format "application/voucher-cose+cbor" in its voucher request to MASA, when the Pledge uses this format in its requests to the Registrar [RFC8995].

The MASA only needs to support formats for which there are Pledges that use that format.

The Registrar MUST use the same format for the RVR as the Pledge used for its PVR.

The Registrar indicates the voucher format it wants to receive from MASA using the HTTP Accept header. This format MUST be the same as the format of the PVR, so that the Pledge can parse it.

At the moment of writing the creation of coaps based MASAs is deemed unrealistic. The use of CoAP for the BRSKI-MASA connection can be the subject of another document. Some consideration was made to specify CoAP support for consistency, but:

- * the Registrar is not expected to be so constrained that it cannot support HTTPS client connections.
- * the technology and experience to build Internet-scale HTTPS responders (which the MASA is) is common, while the experience doing the same for CoAP is much less common.
- * a Registrar is likely to provide onboarding services to both constrained and non-constrained devices. Such a Registrar would need to speak HTTPS anyway.
- * a manufacturer is likely to offer both constrained and non-constrained devices, so there may in practice be no situation in which the MASA could be CoAP-only. Additionally, as the MASA is intended to be a function that can easily be outsourced to a third-party service provider, reducing the complexity would also seem to reduce the cost of that function.
- * security-related considerations: see Section 14.6.

7.2. Registrar Voucher Request

If the PVR contains a proximity assertion, the Registrar MUST propagate this assertion into the RVR by including the "assertion" field with the value "proximity". This conforms to the example in Section 3.3 of [RFC8995] of carrying the assertion forward.

7.3. MASA and the Server Name Indicator (SNI)

A TLS/HTTPS connection is established between the Registrar and MASA.

Section 5.4 of [RFC8995] explains this process, and there are no externally visible changes. A MASA that supports the unconstrained voucher formats should be able to support constrained voucher formats equally well.

There is no requirement that a single MASA be used for both constrained and unconstrained voucher requests: the choice of MASA is determined by the id-mod-MASAURLExtn2016 extension contained in the IDDevID.

The Registrar MUST do [RFC6125] DNS-ID checks on the contents of the certificate provided by the MASA.

In contrast to the Pledge/Registrar situation, the Registrar always knows the name of the MASA, and MUST always include an [RFC6066] Server Name Indicator. The SNI is optional in TLS1.2, but common. The SNI is considered mandatory with TLS1.3.

The presence of the SNI is needed by the MASA, in order for the MASA's server to host multiple tenants (for different customers).

The Registrar SHOULD use a TLS Client Certificate to authenticate to the MASA per Section 5.4.1 of [RFC8995]. If the certificate that the Registrar uses is marked as a id-kp-cmcRA certificate, via Extended Key Usage, then it MUST also have the id-kp-clientAuth EKU attribute set.

7.3.1. Registrar Certificate Requirement

In summary for typical Registrar use, where a single Registrar certificate is used, then the certificate MUST have EKU of: id-kp-cmcRA, id-kp-serverAuth, id-kp-clientAuth.

8. Pinning in Voucher Artifacts

The voucher is a statement by the MASA for use by the Pledge that provides the identity of the Pledge's owner. This section describes how the owner's identity is determined and how it is specified within the voucher.

8.1. Registrar Identity Selection and Encoding

Section 5.5 of [RFC8995] describes BRSKI policies for selection of the owner identity. It indicates some of the flexibility that is possible for the Registrar, and recommends the Registrar to include only certificates in the voucher request (CMS) signing structure that participate in the certificate chain that is to be pinned.

The MASA is expected to evaluate the certificates included by the Registrar in its voucher request, forming them into a chain with the Registrar's (signing) identity on one end. Then, it pins a certificate selected from the chain. For instance, for a domain with a two-level certification authority (see Figure 1), where the voucher-request has been signed by "Registrar", its signing structure includes two additional CA certificates. The arrows in the figure indicate the issuing of a certificate, i.e. author of (1) issued (2) and author of (2) issued (3).

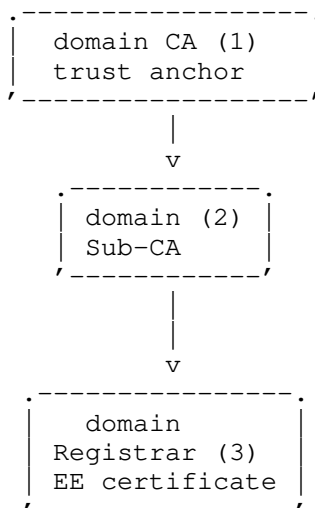


Figure 1: Two-Level CA PKI

When the Registrar is using a COSE-signed constrained voucher request towards MASA, instead of a regular CMS-signed voucher request, the COSE_Sign1 object contains a protected and an unprotected header. The Registrar MUST place all the certificates needed to validate the signature chain from the Registrar on the RVR in an "x5bag" attribute in the unprotected header [I-D.ietf-cose-x509].

The "x5bag" attribute is very important as it provides the required signals from the Registrar to control what identity is pinned in the resulting voucher. This is explained in the next section.

8.2. MASA Pinning Policy

The MASA, having assembled and verified the chain in the signing structure of the voucher request needs to select a certificate to pin. (For the case that only the Registrar's End-Entity certificate is included, only this certificate can be selected and this section does not apply.) The BRSKI policy for pinning by the MASA as described in Section 5.5.2 of [RFC8995] leaves much flexibility to the manufacturer.

The present document adds the following rules to the MASA pinning policy to reduce the network load:

1. for a voucher containing a nonce, it SHOULD select the most specific (lowest-level) CA certificate in the chain.
2. for a nonceless voucher, it SHOULD select the least-specific (highest-level) CA certificate in the chain that is allowed under the MASA's policy for this specific domain.

The rationale for 1. is that in case of a voucher with nonce, the voucher is valid only in scope of the present DTLS connection between Pledge and Registrar anyway, so there is no benefit to pin a higher-level CA. By pinning the most specific CA the constrained Pledge can validate its DTLS connection using less crypto operations. The rationale for pinning a CA instead of the Registrar's End-Entity certificate directly is based on the following benefit on constrained networks: the pinned certificate in the voucher can in common cases be re-used as a Domain CA trust anchor during the EST enrollment and during the operational phase that follows after EST enrollment, as explained in Section 6.6.1.

The rationale for 2. follows from the flexible BRSKI trust model for, and purpose of, nonceless vouchers (Sections 5.5.* and 7.4.1 of [RFC8995]).

Referring to Figure 1 of a domain with a two-level certification authority, the most specific CA ("Sub-CA") is the identity that is pinned by MASA in a nonced voucher. A Registrar that wished to have only the Registrar's End-Entity certificate pinned would omit the "domain CA" and "Sub-CA" certificates from the voucher-request.

In case of a nonceless voucher, depending on the trust level, the MASA pins the "Registrar" certificate (low trust in customer), or the "Sub-CA" certificate (in case of medium trust, implying that any Registrar of that sub-domain is acceptable), or even the "domain CA" certificate (in case of high trust in the customer, and possibly a pre-agreed need of the customer to obtain flexible long-lived vouchers).

8.3. Pinning of Raw Public Keys

Specifically for constrained use cases, the pinning of the raw public key (RPK) of the Registrar is also supported in the constrained voucher, instead of an X.509 certificate. If an RPK is pinned it MUST be the RPK of the Registrar.

When the Pledge is known by MASA to support RPK but not X.509 certificates, the voucher produced by the MASA pins the RPK of the Registrar in either the "pinned-domain-pubk" or "pinned-domain-pubk-sha256" field of a voucher. This is described in more detail in Section 9.2.3. A Pledge that does not support X.509 certificates cannot use EST to enroll; it has to use another method for enrollment without certificates and the Registrar has to support this method also. It is possible that the Pledge will not enroll, but instead only a network join operation will occur (See [RFC9031]). How the Pledge discovers this method and details of the enrollment method are out of scope of this document.

When the Pledge is known by MASA to support PKIX format certificates, the "pinned-domain-cert" field present in a voucher typically pins a domain certificate. That can be either the End-Entity certificate of the Registrar, or the certificate of a domain CA of the Registrar's domain as specified in Section 8.2. However, if the Pledge is known to also support RPK pinning and the MASA intends to identify the Registrar in the voucher (not the CA), then MASA MUST pin the RPK (RPK3 in Figure 2) of the Registrar instead of the Registrar's End-Entity certificate to save space in the voucher.

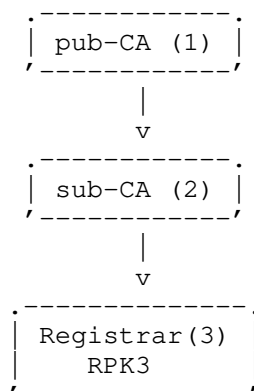


Figure 2: Raw Public Key pinning

8.4. Considerations for use of IDevID-Issuer

[RFC8366] and [RFC8995] defines the idevid-issuer attribute for voucher and voucher-request (respectively), but they summarily explain when to use it.

The use of idevid-issuer is provided so that the serial-number to which the issued voucher pertains can be relative to the entity that issued the devices' IDevID. In most cases there is a one to one relationship between the trust anchor that signs vouchers (and is trusted by the pledge), and the Certification Authority that signs the IDevID. In that case, the serial-number in the voucher must refer to the same device as the serial-number that is in IDevID certificate.

However, there situations where the one to one relationship may be broken. This occurs whenever a manufacturer has a common MASA, but different products (on different assembly lines) are produced with identical serial numbers. A system of serial numbers which is just a simple counter is a good example of this. A system of serial numbers where there is some prefix relating the product type does not fit into this, even if the lower digits are a counter.

It is not possible for the Pledge or the Registrar to know which situation applies. The question to be answered is whether or not to include the idevid-issuer in the PVR and the RVR. A second question arises as to what the format of the idevid-issuer contents are.

Analysis of the situation shows that the pledge never needs to include the idevid-issuer in its PVR, because the pledge's IDevID certificate is available to the Registrar, and the Authority Key Identifier is contained within that. The pledge therefore has no need to repeat this.

For the RVR, the Registrar has to examine the pledge's IDevID certificate to discover the serial number for the Registrar's Voucher Request (RVR). This is clear in Section 5.5 of [RFC8995]. That section also clarifies that the idevid-issuer is to be included in the RVR.

Concerning the Authority Key Identifier, [RFC8366] specifies that the entire object i.e. the extnValue OCTET STRING is to be included: comprising the AuthorityKeyIdentifier, SEQUENCE, Choice as well as the OCTET STRING that is the keyIdentifier.

9. Artifacts

This section describes for both the voucher request and the voucher first the abstract (tree) definition as explained in [RFC8340]. 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-sid].

9.1. Voucher Request artifact

9.1.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with the addition of the fields proximity-registrar-pubk, proximity-registrar-pubk-sha256, proximity-registrar-cert, and prior-signed-voucher-request.

prior-signed-voucher-request is only used between the Registrar and the MASA. proximity-registrar-pubk or proximity-registrar-pubk-sha256 optionally replaces proximity-registrar-cert for the most constrained cases where RPK is used by the Pledge.

```
module: ietf-voucher-request-constrained
```

```
  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-pubk?  binary
      +-- proximity-registrar-pubk-sha256? binary
      +-- proximity-registrar-cert?  binary
      +-- prior-signed-voucher-request? binary
```

9.1.2. SID values

```
  SID Assigned to
```

```
-----
2501 data /ietf-voucher-request-constrained:voucher
2502 data .../assertion
2503 data .../created-on
2504 data .../domain-cert-revocation-checks
2505 data .../expires-on
2506 data .../idevid-issuer
2507 data .../last-renewal-date
2508 data /ietf-voucher-request-constrained:voucher/nonce
2509 data .../pinned-domain-cert
2510 data .../prior-signed-voucher-request
2511 data .../proximity-registrar-cert
2513 data .../proximity-registrar-pubk
2512 data .../proximity-registrar-pubk-sha256
2514 data .../serial-number
```

WARNING, obsolete definitions

The "assertion" attribute is an enumerated type [RFC8366], and the current PYANG tooling does not document the valid values for this attribute. In the JSON serialization, the literal strings from the enumerated types are used so there is no ambiguity. In the CBOR serialization, a small integer is used. This following values are documented here, but the YANG module should be considered authoritative. No IANA registry is provided or necessary because the YANG module provides for extensions.

Integer	Assertion Type
0	verified
1	logged
2	proximity

Table 3: CBOR integers
for the "assertion"
attribute enum

9.1.3. YANG Module

In the voucher-request-constrained YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the voucher-request-constrained module name, all voucher attributes, and the voucher-request-constrained attributes. Two attributes of the voucher are "refined" to be optional.

```
<CODE BEGINS> file "ietf-voucher-request-constrained@2021-04-15.yang"
module ietf-voucher-request-constrained {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-request-constrained";
  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 its 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
{[I-D.ietf-anima-bootstrap-keyinfra]}

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 either a pinned Raw Public Key of the Registrar, or by a pinned X.509 certificate of the Registrar or domain CA.

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 "2021-04-15" {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: Voucher Profile for Constrained Devices";  
}
```

```
rc: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-pubk {
      type binary;
      description
        "The proximity-registrar-pubk replaces
        the proximity-registrar-cert in constrained uses of
        the voucher-request.
        The proximity-registrar-pubk 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, but due to
        size is discouraged.";
    }

    leaf proximity-registrar-pubk-sha256 {
      type binary;
      description
        "The proximity-registrar-pubk-sha256
        is an alternative to both
        proximity-registrar-pubk and pinned-domain-cert.
        In many cases the public key of the domain has already
        been transmitted during the key agreement protocol,
        and it is wasteful to transmit the public key another
        two times.
        The use of a hash of public key info, at 32-bytes for
        sha256 is a significant savings compared to an RSA
        public key, but is only a minor savings compared to
        a 256-bit ECDSA public-key."
    }
  }
}
```

```
        Algorithm agility is provided by extensions to this
        specification which may define a new leaf for another
        hash type.";
    }

    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 field or one of its
            alternatives 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.";
    }
}
```

```

    }
  }
}
<CODE ENDS>

```

9.1.4. Example voucher request artifact

Below is a CBOR serialization of an example constrained voucher request from a Pledge to a Registrar, shown in CBOR diagnostic notation. The enum value of the assertion field is calculated to be 2 by following the algorithm described in section 9.6.4.2 of [RFC7950].

```

{
  2501: {
    +2 : "2016-10-07T19:31:42Z", / SID=2503, created-on /
    +4 : "2016-10-21T19:31:42Z", / SID=2505, expires-on /
    +1 : 2, / SID=2502, assertion "proximity" /
    +13: "JADA123456789", / SID=2514, serial-number /
    +5 : h'08C2BF36....B3D2B3', / SID=2506, idevid-issuer /
    +10: h'30820275....82c35f', / SID=2511, proximity-registrar-cert/
    +3 : true, / SID=2504, domain-cert
              -revocation-checks/
    +6 : "2017-10-07T19:31:42Z" / SID=2507, last-renewal-date /
  }
}

```

9.2. 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.

9.2.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with only the addition of the fields pinned-domain-pubk and pinned-domain-pubk-sha256.

module: ietf-voucher-constrained

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-pubk?  binary
  +-- pinned-domain-pubk-sha256? binary

```

9.2.2. SID values

SID Assigned to

```

-----
2451 data /ietf-voucher-constrained:voucher
2452 data /ietf-voucher-constrained:voucher/assertion
2453 data /ietf-voucher-constrained:voucher/created-on
2454 data .../domain-cert-revocation-checks
2455 data /ietf-voucher-constrained:voucher/expires-on
2456 data /ietf-voucher-constrained:voucher/idevid-issuer
2457 data .../last-renewal-date
2458 data /ietf-voucher-constrained:voucher/nonce
2459 data .../pinned-domain-cert
2460 data .../pinned-domain-pubk
2461 data .../pinned-domain-pubk-sha256
2462 data /ietf-voucher-constrained:voucher/serial-number

```

WARNING, obsolete definitions

The "assertion" enumerated attribute is numbered as per Section 9.1.2.

9.2.3. YANG Module

In the voucher-constrained YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the voucher-constrained module name, all voucher attributes, and the voucher-constrained attributes. Two attributes of the voucher are "refined" to be optional.

```
<CODE BEGINS> file "ietf-voucher-constrained@2021-04-15.yang"
module ietf-voucher-constrained {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-constrained";
  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 its delegate
     (MASA) to securely assign one or more pledges to an 'owner',
     so that a pledge 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 either a pinned Raw Public Key of the Registrar, or by a
     pinned X.509 certificate of the Registrar or domain CA."
```

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 "2021-04-15" {
  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-pubk {
        type binary;
        description
          "The pinned-domain-pubk may replace the
           pinned-domain-cert in constrained uses of
           the voucher. The pinned-domain-pubk
           is the Raw Public Key of the Registrar.
           This field is encoded as a Subject Public Key Info block
           as specified in RFC7250, in 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 pinned-domain-pubk-sha256 {
        type binary;
        description
            "The pinned-domain-pubk-sha256 is a second
            alternative to pinned-domain-cert. In many cases the
            public key of the domain has already been transmitted
            during the key agreement process, and it is wasteful
            to transmit the public key another two times.
            The use of a hash of public key info, at 32-bytes for
            sha256 is a significant savings compared to an RSA
            public key, but is only a minor savings compared to
            a 256-bit ECDSA public-key.
            Algorithm agility is provided by extensions to this
            specification which can define a new leaf for another
            hash type.";
    }
}
}
}
}
}
<CODE ENDS>

```

9.2.4. Example voucher artifacts

Below the CBOR serialization of an example constrained voucher is shown in CBOR diagnostic 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].

```

{
  2451: {
    +2 : "2016-10-07T19:31:42Z", / SID = 2453, created-on /
    +4 : "2016-10-21T19:31:42Z", / SID = 2455, expires-on /
    +1 : 0, / SID = 2452, assertion "verified" /
    +11: "JADA123456789", / SID = 2462, serial-number /
    +5 : h'E40393B4....68A3', / SID = 2456, idevid-issuer /
    +8 : h'30820275....C35F', / SID = 2459, pinned-domain-cert/
    +3 : true, / SID = 2454, domain-cert /
    / -revocation-checks /
    +6 : "2017-10-07T19:31:42Z" / SID = 2457, last-renewal-date /
  }
}

```

9.3. Signing voucher and voucher-request artifacts with COSE

The COSE_Sign1 structure is discussed in Section 4.2 of [I-D.ietf-cose-rfc8152bis-struct]. 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.

Support for ECDSA with SHA2-256 using curve secp256r1 (aka prime256k1) is RECOMMENDED. Most current low power hardware has support for acceleration of this algorithm. Future hardware designs could omit this in favour of a newer algorithms. This is the ES256 keytype from Table 1 of [I-D.ietf-cose-rfc8152bis-algs]. Support for curve secp256k1 is OPTIONAL.

Support for EdDSA using Curve 25519 is RECOMMENDED in new designs if hardware support is available. This is keytype EDDSA (-8) from Table 2 of [I-D.ietf-cose-rfc8152bis-algs]. A "crv" parameter is necessary to specify the Curve, which from Table 18. The 'kty' field MUST be present, and it MUST be 'OKP'. (Table 17)

A transition towards EdDSA is occurring in the industry. Some hardware can accelerate only some algorithms with specific curves, other hardware can accelerate any curve, and still other kinds of hardware provide a tool kit for acceleration of any elliptic curve algorithm.

In general, the Pledge is expected to support only a single algorithm, while the Registrar, usually not constrained, is expected to support a wide variety of algorithms: both legacy ones and up-and-coming ones via regular software updates.

An example of the supported COSE_Sign1 object structure is shown in Figure 3.

```
18( / COSE_Sign1 /
  [
    h'A101382E',          / protected header encoding: {1: -47}      /
    {                    /      which means { "alg": ES256K }      /
      4 : h'7890A03F1234' / 4 is the "kid" binary key identifier /
    },
    h'1234....5678', / voucher-request binary content (CBOR)      /
    h'4567....1234' / voucher-request binary public signature     /
  ]
)
```

Figure 3: COSE_Sign1 example in CBOR diagnostic notation

The [COSE-registry] specifies the integers/encoding for the "alg" and "kid" fields in Figure 3. The "alg" field restricts the key usage for verification of this COSE object to a particular cryptographic algorithm.

The "kid" field is optionally present: it is an unprotected field that identifies the public key of the key pair that was used to sign this message. The value of the key identifier "kid" parameter is an example value. Usually a hash of the public key is used to identify the public key, but a device serial number may also be used. The choice of key identifier method is vendor-specific. If "kid" is not present, then a verifying party needs to use other context information to retrieve the right public key to verify the COSE_Sign1 object against. For example, this context information may be a unique serial number encoded in the binary content (CBOR) field.

A Registrar MAY use a "kid" parameter in its RVR to identify its signing key as used to sign the RVR. The method of generating this "kid" is vendor-specific and SHOULD be configurable in the Registrar to support commonly used methods. In order to support future business cases and supply chain integrations, a Registrar MUST be configurable, on a per-manufacturer basis, to be able to configure the "kid" to a particular value. Both binary and string values are to be supported, each being inserted using a CBOR bstr or tstr. By default, a Registrar does not include a "kid" parameter in its RVR since the signing key is already identified by the included signing certificates in the COSE "x5bag" structure.

A Pledge normally SHOULD NOT use a "kid" parameter in its PVR, because its signing key is already identified by the Pledge's unique serial number that is included in the PVR. Still, where needed the Pledge MAY use a "kid" parameter in its PVR to help the MASA identify the right public key to verify against. This can occur for example if a Pledge has multiple IDevID identities. A Registrar normally SHOULD ignore a "kid" parameter used in a received PVR, as this information is intended for the MASA. In other words, there is no need for the Registrar to verify the contents of this field, but it may include it in an audit log.

In Appendix C a binary COSE_Sign1 object is shown based on the voucher-request example of Section 9.1.4.

10. Deployment-specific Discovery Considerations

This section details how discovery is done in specific deployment scenarios.

10.1. 6TSCH Deployments

In 6TISCH networks, the Constrained Join Proxy (CoJP) mechanism is described in [RFC9031]. Such networks are expected to use a [I-D.ietf-lake-edhoc] to do key management. This is the subject of future work. The Enhanced Beacon has been extended in [RFC9032] to allow for discovery of the Join Proxy.

10.2. Generic networks using GRASP

[RFC8995] defines a mechanism for the Pledge to discover a Join Proxy by listening for [RFC8990] GRASP messages. This mechanism can be used on any network which does not have another more specific mechanism. This mechanism supports mesh networks, and can also be used over unencrypted WIFI.

10.3. Generic networks using mDNS

[RFC8995] also defines a non-normative mechanism for the Pledge to discover a Join Proxy by doing mDNS queries. This mechanism can be used on any network which does not have another recommended mechanism. This mechanism does not easily support mesh networks. It can be used over unencrypted WIFI.

10.4. Thread networks using Mesh Link Establishment (MLE)

Thread [Thread] is a wireless mesh network protocol based on 6LoWPAN [RFC6282] and other IETF protocols. In Thread, a new device discovers potential Thread networks and Thread routers to join by using the Mesh Link Establishment (MLE) [I-D.ietf-6lo-mesh-link-establishment] protocol. MLE uses the UDP port number 19788. The new device sends discovery requests on different IEEE 802.15.4 radio channels, to which routers (if any present) respond with a discovery response containing information about their respective network. Once a suitable router is selected the new device initiates a DTLS transport-layer secured connection to the network's commissioning application, over a link-local single radio hop to the selected Thread router. This link is not yet secured at the radio level: link-layer security will be set up once the new device is approved by the commissioning application to join the Thread network, and it gets provisioned with network access credentials.

The Thread router acts here as a Join Proxy. The MLE discovery response message contains UDP port information to signal the new device which port to use for its DTLS connection.

10.5. Non-mesh networks using CoAP Discovery

On unencrypted constrained networks such as 802.15.4, CoAP discover may be done using the mechanism detailed in [I-D.ietf-ace-coap-est] section 5.1.

11. Design Considerations

The design considerations for the CBOR encoding of vouchers are much the same as for JSON vouchers in [RFC8366]. One key difference is that the names of the leaves in the YANG definition do not affect the size of the resulting CBOR, as the SID translation process assigns integers to the names.

Any POST request to the Registrar with resource /vs or /es returns a 2.04 Changed response with empty payload. The client should be aware that the server may use a piggybacked CoAP response (ACK, 2.04) but may also respond with a separate CoAP response, i.e. first an (ACK, 0.0) that is an acknowledgement of the request reception followed by a (CON, 2.04) response in a separate CoAP message.

12. Raw Public Key Use Considerations

This section explains techniques to reduce the number of bytes that are sent over the wire during the BRSKI bootstrap. The use of a raw public key (RPK) in the pinning process can significantly reduce the number of bytes and round trips, but it comes with a few significant operational limitations.

12.1. The Registrar Trust Anchor

When the Pledge first connects to the Registrar, the connection to the Registrar is provisional, as explained in Section 5.6.2 of [RFC8995]. The Registrar provides its public key in a TLSServerCertificate, and the Pledge uses that to validate that integrity of the (D)TLS connection, but it does not validate the identity of the provided certificate.

As the TLSServerCertificate object is never verified directly by the pledge, sending it can be considered superfluous. Instead of using a (TLSServer)Certificate of type X509 (see section 4.4.2 of [RFC8446]), a RawPublicKey object is used.

A Registrar operating in a mixed environment can determine whether to send a Certificate or a Raw Public key: this is determined by the pledge including the server_certificate_type of RawPublicKey. This is shown in section 5 of [RFC7250].

The Pledge continues to send a `client_certificate_type` of X509, so that the Registrar can properly identify the pledge and distill the MASA URI information from its certificate.

12.2. The Pledge Voucher Request

The Pledge puts the Registrar's public key into the `proximity-registrar-pubk` field of the `voucher-request`. (The `proximity-registrar-pubk-sha256` can also be used if the 32-bytes of a SHA256 hash turns out to be smaller than a typical ECDSA key.)

As the format of the `pubk` field is identical to the TLS Certificate `RawPublicKey`, no manipulation at all is needed to insert this into a `voucher-request`.

12.3. The Voucher Response

A returned voucher will have a `pinned-domain-subk` field with the identical key as was found in the `proximity-registrar-pubk` field above, as well as in the TLS connection.

Validation of this key by the pledge is what takes the DTLS connection out of the provisional state see Section 5.6.2 of [RFC8995].

The voucher needs to be validated first. The Pledge needs to have a public key to validate the signature from the MASA on the voucher.

In certain cases, the MASA's public key counterpart of the (private) signing key is already installed in the Pledge at manufacturing time. In other cases, if the MASA signing key is based upon a PKI (see [I-D.richardson-anima-masa-considerations] Section 2.3), then a certificate chain may need to be included with the voucher in order for the pledge to validate the signature. In CMS signed artifacts, the CMS structure has a place for such certificates.

In the COSE-signed Constrained Vouchers described in this document, the `x5bag` attribute from [I-D.ietf-cose-x509] is to be used for this.

13. Use of constrained vouchers with HTTPS

This specification contains two extensions to [RFC8995]: a constrained voucher format (COSE), and a constrained transfer protocol (CoAP).

On constrained networks with constrained devices, it make senses to use both together. However, this document does not mandate that this be the only way.

A given constrained device design and software may be re-used for multiple device models, such as a model having only an IEEE 802.15.4 radio, or a model having only an IEEE 802.11 (Wi-Fi) radio, or a model having both these radios. A manufacturer of such device models may wish to have code only for the use of the constrained voucher format (COSE), and use it on all supported radios including the IEEE 802.11 radio. For this radio, the software stack to support HTTP/TLS may be already integrated into the radio module hence it is attractive for the manufacturer to reuse this. This type of approach is supported by this document. In the case that HTTPS is used, the normal [RFC8995] resource names are used, together with the media types described in this document.

Other combinations are possible, but they are not enumerated here. New work such as [I-D.ietf-anima-jws-voucher] provides new formats that may be useable over a number of different transports. In general, sending larger payloads over constrained networks makes less sense, while sending smaller payloads over unconstrained networks is perfectly acceptable.

The Pledge will in most cases support a single voucher format, which it uses without negotiation i.e. without discovery of formats supported. The Registrar, being unconstrained, is expected to support all voucher formats. There will be cases where a Registrar does not support a new format that a new Pledge uses, and this is an unfortunate situation that will result in lack of interoperation.

The responsibility for supporting new formats is on the Registrar.

14. Security Considerations

14.1. Duplicate serial-numbers

In the absense of correct use of idevid-issuer by the Registrar as detailed in Section 8.4, it would be possible for a malicious Registrar to use an unauthorized voucher for a device. This would apply only to the case where a Manufacturer Authorized Signing Authority (MASA) is trusted by different products from the same manufacturer, and the manufacturer has duplicated serial numbers as a result of a merge, acquisition or mis-management.

For example, imagine the same manufacturer makes light bulbs as well as gas centrifuges, and said manufacturer does not uniquely allocate product serial numbers. This attack only works for nonceless vouchers. The attacker has obtained a light bulb which happens to have the same serial-number as a gas centrifuge which it wishes to obtain access. The attacker performs a normal BRSKI onboarding for the light bulb, but then uses the resulting voucher to onboard the

gas centrifuge. The attack requires that the gas centrifuge be returned to a state where it is willing to perform a new onboarding operation.

This attack is prevented by the mechanism of having the Registrar include the idevid-issuer in the RVR, and the MASA including it in the resulting voucher. The idevid-issuer is not included by default: a MASA needs to be aware if there are parts of the organization which duplicates serial numbers, and if so, include it.

14.2. IDevID security in Pledge

The security of this protocol depends upon the Pledge identifying itself to the Registrar using its manufacturer installed certificate: the IDevID certificate. Associated with this certificate is the IDevID private key, known only to the Pledge. Disclosure of this private key to an attacker would permit the attacker to impersonate the Pledge towards the Registrar, probably gaining access credentials to that Registrar's network.

If the IDevID private key disclosure is known to the manufacturer, there is little recourse other than recall of the relevant part numbers. The process for communicating this recall would be within the BRSKI-MASA protocol. Neither this specification nor [RFC8995] provides for consultation of a Certification Revocation List (CRL) or Open Certificate Status Protocol (OCSP) by a Registrar when evaluating an IDevID certificate. However, the BRSKI-MASA protocol submits the IDevID from the Registrar to the manufacturer's MASA and a manufacturer would have an opportunity to decline to issue a voucher for a device which they believe has become compromised.

It may be difficult for a manufacturer to determine when an IDevID private key has been disclosed. Two situations present themselves: in the first situation a compromised private key might be reused in a counterfeit device, which is sold to another customer. This would present itself as an onboarding of the same device in two different networks. The manufacturer may become suspicious seeing two voucher requests for the same device from different Registrars. Such activity could be indistinguishable from a device which has been resold from one operator to another, or re-deployed by an operator from one location to another.

In the second situation, an attacker having compromised the IDevID private key of a device might then install malware into the same device and attempt to return it to service. The device, now blank, would go through a second onboarding process with the original Registrar. Such a Registrar could notice that the device has been "factory reset" and alert the operator to this situation. One remedy

against the presence of malware is through the use of Remote Attestation such as described in [I-D.ietf-rats-architecture]. Future work will need to specify a background-check Attestation flow as part of the voucher-request/voucher-response process. Attestation may still require access to a private key (e.g. IDevID private key) in order to sign Evidence, so a primary goal should be to keep any private key safe within the Pledge.

In larger, more expensive, systems there is budget (power, space, and bill of materials) to include more specific defenses for a private key. For instance, this includes putting the IDevID private key in a Trusted Platform Module (TPM), or use of Trusted Execution Environments (TEE) for access to the key. On smaller IoT devices, the cost and power budget for an extra part is often prohibitive.

It is becoming more and more common for CPUs to have an internal set of one-time fuses that can be programmed (often they are "burnt" by a laser) at the factory. This section of memory is only accessible in some privileged CPU state. The use of this kind of CPU is appropriate as it provides significant resistance against key disclosure even when the device can be disassembled by an attacker.

In a number of industry verticals, there is increasing concern about counterfeit parts. These may be look-alike parts created in a different factory, or parts which are created in the same factory during an illegal night-shift, but which are not subject to the appropriate level of quality control. The use of a manufacturer-signed IDevID certificate provides for discovery of the pedigree of each part, and this often justifies the cost of the security measures associated with storing the private key.

14.3. Security of CoAP and UDP protocols

Section 7.1 explains that no CoAPS version of the BRSKI-MASA protocol is proposed. The connection from the Registrar to the MASA continues to be HTTPS as in [RFC8995]. This has been done to simplify the MASA deployment for the manufacturer, because no new protocol needs to be enabled on the server.

The use of UDP protocols across the open Internet is sometimes fraught with security challenges. Denial-of-service attacks against UDP based protocols are trivial as there is no three-way handshake as done for TCP. The three-way handshake of TCP guarantees that the node sending the connection request is reachable using the origin IP address. While DTLS contains an option to do a stateless challenge -- a process actually stronger than that done by TCP -- it is not yet common for this mechanism to be available in hardware at multigigabit speeds. It is for this reason that this document defines using HTTPS for the Registrar to MASA connection.

14.4. Registrar Certificate may be self-signed

The provisional (D)TLS connection formed by the Pledge with the Registrar does not authenticate the Registrar's identity. This Registrar's identity is validated by the [RFC8366] voucher that is issued by the MASA, signed with an anchor that was built-in to the Pledge.

The Registrar may therefore use any certificate, including a self-signed one. The only restrictions on the certificate is that it **MUST** have EKU bits set as detailed in Section 7.3.1.

14.5. Use of RPK alternatives to proximity-registrar-cert

In Section 9.1 two compact alternative fields for proximity-registrar-cert are defined that include an RPK: proximity-registrar-pubk and proximity-registrar-pubk-sha256. The Pledge can use these fields in its PVR to identify the Registrar based on its public key only. Since the full certificate of the proximate Registrar is not included, use of these fields by a Pledge implies that a Registrar could insert another certificate with the same public key identity into the RVR. For example, an older or a newer version of its certificate. The MASA will not be able to detect such act by the Registrar. But since any 'other' certificate the Registrar could insert in this way still encodes its identity the additional risk of using the RPK alternatives is negligible.

When a Registrar sees a PVR that uses one of proximity-registrar-pubk or proximity-registrar-pubk-sha256 fields, this implies the Registrar must include the certificate identified by these fields into its RVR. Otherwise, the MASA is unable to verify proximity. This requirement is already implied by the "MUST" requirement in Section 8.1.

14.6. MASA support of CoAPS

The use of CoAP for the BRSKI-MASA connection is not in scope of the current document. The following security considerations have led to this choice of scope:

- * the technology and experience to build secure Internet-scale HTTPS responders (which the MASA is) is common, while the experience in doing the same for CoAP is much less common.
- * in many enterprise networks, outgoing UDP connections are often treated as suspicious, which could effectively block CoAP connections for some firewall configurations.
- * reducing the complexity of MASA (i.e. less protocols supported) would also reduce its potential attack surface, which is relevant since the MASA is 24/7 exposed on the Internet and accepting (untrusted) incoming connections.

15. IANA Considerations

15.1. Resource Type Registry

Additions to the sub-registry "Resource Type Link Target Attribute Values", within the "CoRE Parameters" IANA registry are specified below.

Reference: [This RFC]

Attribute	Description
brski	Root path of Bootstrapping Remote Secure Key Infrastructure (BRSKI) resources
brski.rv	BRSKI request voucher resource
brski.vs	BRSKI voucher status telemetry resource
brski.es	BRSKI enrollment status telemetry resource

Table 4: Resource Type (rt) link target attribute values for IANA registration

15.2. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-voucher-constrained
 Registrant Contact: The ANIMA WG of the IETF.
 XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-voucher-request-constrained
 Registrant Contact: The ANIMA WG of the IETF.
 XML: N/A, the requested URI is an XML namespace.

15.3. 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 the following registration is requested:

name: ietf-voucher-constrained
 namespace: urn:ietf:params:xml:ns:yang:ietf-voucher-constrained
 prefix: vch
 reference: RFC XXXX

name: ietf-voucher-request-constrained
 namespace: urn:ietf:params:xml:ns:yang:ietf-voucher-request-constrained
 prefix: vch
 reference: RFC XXXX

15.4. The RFC SID range assignment sub-registry

Entry-point	Size	Module name	RFC Number
2450	50	ietf-voucher-constrained	[This RFC]
2500	50	ietf-voucher-request -constrained	[This RFC]

Warning: These SID values are defined in [I-D.ietf-core-sid], not as an Early Allocation.

IANA: please update the names in the Registry to match these revised names, if they have not already been revised.

15.5. Media Types Registry

This section registers the 'application/voucher-cose+cbor' in the IANA "Media Types" registry. This media type is used to indicate that the content is a CBOR voucher or voucher request signed with a COSE_Sign1 structure [I-D.ietf-cose-rfc8152bis-struct].

15.5.1. application/voucher-cose+cbor

Type name: application
Subtype name: voucher-cose+cbor
Required parameters: N/A
Optional parameters: N/A
Encoding considerations: binary (CBOR)
Security considerations: Security Considerations of [This RFC].
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 onboarding systems
Fragment identifier considerations: The syntax and semantics of fragment identifiers specified for application/voucher-cose+cbor are as specified for application/cbor. (At publication of this document, there is no fragment identification syntax defined for application/cbor.)
Additional information:
 Deprecated alias names for this type: N/A
 Magic number(s): N/A
 File extension(s): .vch
 Macintosh file type code(s): N/A
Person & email address to contact for further information: IETF ANIMA Working Group (anima@ietf.org) or IETF Operations and Management Area Working Group (opsawg@ietf.org)
Intended usage: COMMON
Restrictions on usage: N/A
Author: ANIMA WG
Change controller: IETF
Provisional registration? (standards tree only): NO

15.6. CoAP Content-Format Registry

One addition to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry is needed for a new content-format. It can be registered in the Expert Review range (0-255) or the IETF Review range (256-9999).

Media type	Encoding	ID	Reference
application/voucher-cose+cbor	-	TBD3	[This RFC]

15.7. Update to BRSKI Parameters Registry

This section updates the BRSKI Well-Known URIs sub-registry of the IANA Bootstrapping Remote Secure Key Infrastructures (BRSKI) Parameters Registry by adding a new column "Short URI". The contents of this field MUST be specified for any newly registered URI as follows:

Short URI: A short name for the "URI" resource that can be used by a Constrained BRSKI Pledge in a CoAP request to the Registrar. In case the "URI" resource is only used between Registrar and MASA, the value "--" is registered denoting that a short name is not applicable.

The initial contents of the sub-registry including the new column are as follows:

URI	Short URI	Description	Reference
requestvoucher	rv	Request voucher: Pledge to Registrar, and Registrar to MASA	[RFC8995], [This RFC]
voucher_status	vs	Voucher status telemetry: Pledge to Registrar	[RFC8995], [This RFC]
requestauditlog	--	Request audit log: Registrar to MASA	[RFC8995]
enrollstatus	es	Enrollment status telemetry: Pledge to Registrar	[RFC8995], [This RFC]

Table 5: Update of the BRSKI Well-Known URI Sub-Registry

16. Acknowledgements

We are very grateful to Jim Schaad for explaining COSE and CMS choices. Also thanks to Jim Schaad for correcting earlier versions of the COSE_Sign1 objects.

Michel Veillette did extensive work on `_pyang_` to extend it to support the SID allocation process, and this document was among its first users.

Daniel Franke and Henk Birkholtz provided review feedback.

The BRSKI design team has met on many Thursdays for document review. It includes: Aurelio Schellenbaum, David von Oheimb Steffen Fries, Thomas Werner, Toerless Eckert,

17. Changelog

-11 to -16 (For change details see GitHub issues <https://github.com/anima-wg/constrained-voucher/issues>)

-10 Design considerations extended Examples made consistent

-08 Examples for cose_sign1 are completed and improved.

-06 New SID values assigned; regenerated examples

-04 voucher and request-voucher MUST be signed examples for signed request are added in appendix IANA SID registration is updated SID values in examples are aligned signed cms examples aligned with new SIDs

-03

Examples are inverted.

-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

18. References

18.1. Normative References

[I-D.ietf-ace-coap-est]

Stok, P. V. D., Kampanakis, P., Richardson, M. C., and S. Raza, "EST over secure CoAP (EST-coaps)", Work in Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6 January 2020, <<https://www.ietf.org/archive/id/draft-ietf-ace-coap-est-18.txt>>.

[I-D.ietf-core-sid]

Veillette, M., Pelov, A., Petrov, I., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", Work in Progress, Internet-Draft, draft-ietf-core-sid-18, 18 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-core-sid-18.txt>>.

[I-D.ietf-core-yang-cbor]

Veillette, M., Petrov, I., Pelov, A., Bormann, C., and M. Richardson, "CBOR Encoding of Data Modeled with YANG", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-19, 20 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-core-yang-cbor-19.txt>>.

[I-D.ietf-cose-rfc8152bis-algs]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020, <<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-algs-12.txt>>.

[I-D.ietf-cose-rfc8152bis-struct]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>>.

[I-D.ietf-cose-x509]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", Work in Progress, Internet-Draft, draft-ietf-cose-x509-08, 14 December 2020, <<https://www.ietf.org/archive/id/draft-ietf-cose-x509-08.txt>>.

- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-dtls13-43.txt>>.
- [ieee802-1AR]
IEEE Standard, "IEEE 802.1AR Secure Device Identifier", 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<https://www.rfc-editor.org/info/rfc7250>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.

- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.
- [RFC9031] Vuini, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <<https://www.rfc-editor.org/info/rfc9031>>.
- [RFC9032] Dujovne, D., Ed. and M. Richardson, "Encapsulation of 6TiSCH Join and Enrollment Information Elements", RFC 9032, DOI 10.17487/RFC9032, May 2021, <<https://www.rfc-editor.org/info/rfc9032>>.

18.2. Informative References

- [COSE-registry] IANA, "CBOR Object Signing and Encryption (COSE) registry", 2017, <<https://www.iana.org/assignments/cose/cose.xhtml>>.
- [I-D.ietf-6lo-mesh-link-establishment] Kelsey, R., "Mesh Link Establishment", Work in Progress, Internet-Draft, draft-ietf-6lo-mesh-link-establishment-00, 1 December 2015, <<https://www.ietf.org/archive/id/draft-ietf-6lo-mesh-link-establishment-00.txt>>.
- [I-D.ietf-anima-constrained-join-proxy] Richardson, M., Stok, P. V. D., and P. Kampanakis, "Constrained Join Proxy for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-join-proxy-09, 25 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-constrained-join-proxy-09.txt>>.
- [I-D.ietf-anima-jws-voucher] Richardson, M. and T. Werner, "JWS signed Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-jws-voucher-03, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-jws-voucher-03.txt>>.

- [I-D.ietf-lake-edhoc]
Selander, G., Mattsson, J. P., and F. Palombini,
"Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in
Progress, Internet-Draft, draft-ietf-lake-edhoc-12, 20
October 2021, <<https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-12.txt>>.
- [I-D.ietf-rats-architecture]
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and
W. Pan, "Remote Attestation Procedures Architecture", Work
in Progress, Internet-Draft, draft-ietf-rats-architecture-
15, 8 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-rats-architecture-15.txt>>.
- [I-D.kuehlewind-update-tag]
Kuehlewind, M. and S. Krishnan, "Definition of new tags
for relations between RFCs", Work in Progress, Internet-
Draft, draft-kuehlewind-update-tag-04, 12 July 2021,
<<https://www.ietf.org/archive/id/draft-kuehlewind-update-tag-04.txt>>.
- [I-D.richardson-anima-masa-considerations]
Richardson, M. and W. Pan, "Operational Considerations for
Voucher infrastructure for BRSKI MASA", Work in Progress,
Internet-Draft, draft-richardson-anima-masa-
considerations-06, 13 November 2021,
<<https://www.ietf.org/archive/id/draft-richardson-anima-masa-considerations-06.txt>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet
Control Message Protocol (ICMPv6) for the Internet
Protocol Version 6 (IPv6) Specification", STD 89,
RFC 4443, DOI 10.17487/RFC4443, March 2006,
<<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6
Datagrams over IEEE 802.15.4-Based Networks", RFC 6282,
DOI 10.17487/RFC6282, September 2011,
<<https://www.rfc-editor.org/info/rfc6282>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link
Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
<<https://www.rfc-editor.org/info/rfc6690>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
"Enrollment over Secure Transport", RFC 7030,
DOI 10.17487/RFC7030, October 2013,
<<https://www.rfc-editor.org/info/rfc7030>>.

- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [Thread] Thread Group, Inc, "Thread support page, White Papers", November 2021, <<https://www.threadgroup.org/support#Whitepapers>>.

Appendix A. Library Support for BRSKI

For the implementation of BRSKI, the use of a software library to manipulate certificates and use crypto algorithms is often beneficial. Two C-based examples are OpenSSL and mbedtls. Others more targeted to specific platforms or languages exist. It is important to realize that the library interfaces differ significantly between libraries.

Libraries do not support all known crypto algorithms. Before deciding on a library, it is important to look at their supported crypto algorithms and the roadmap for future support. Apart from availability, the library footprint, and the required execution cycles should be investigated beforehand.

The handling of certificates usually includes the checking of a certificate chain. In some libraries, chains are constructed and verified on the basis of a set of certificates, the trust anchor (usually self signed root CA), and the target certificate. In other libraries, the chain must be constructed beforehand and obey order criteria. Verification always includes the checking of the signatures. Less frequent is the checking the validity of the dates or checking the existence of a revoked certificate in the chain against a set of revoked certificates. Checking the chain on the consistency of the certificate extensions which specify the use of the certificate usually needs to be programmed explicitly.

A library can be used to construct a (D)TLS connection. It is useful to realize that differences between (D)TLS implementations will occur due to the differences in the certificate checks supported by the

library. On top of that, checks between client and server certificates enforced by (D)TLS are not always helpful for a BRSKI implementation. For example, the certificates of Pledge and Registrar are usually not related when the BRSKI protocol is started. It must be verified that checks on the relation between client and server certificates do not hamper a successful DTLS connection establishment.

A.1. OpensSSL

From openssl's apps/verify.c :

```
<CODE BEGINS>
X509 *x = NULL;
int i = 0, ret = 0;
X509_STORE_CTX *csc;
STACK_OF(X509) *chain = NULL;
int num_untrusted;

x = load_cert(file, "certificate file");
if (x == NULL)
    goto end;

csc = X509_STORE_CTX_new();
if (csc == NULL) {
    BIO_printf(bio_err, "error %s: X.509 store context"
               "allocation failed\n",
               (file == NULL) ? "stdin" : file);
    goto end;
}

X509_STORE_set_flags(ctx, vflags);
if (!X509_STORE_CTX_init(csc, ctx, x, uchain)) {
    X509_STORE_CTX_free(csc);
    BIO_printf(bio_err,
               "error %s: X.509 store context"
               "initialization failed\n",
               (file == NULL) ? "stdin" : file);
    goto end;
}
if (tchain != NULL)
    X509_STORE_CTX_set0_trusted_stack(csc, tchain);
if (crls != NULL)
    X509_STORE_CTX_set0_crls(csc, crls);

i = X509_verify_cert(csc);
X509_STORE_CTX_free(csc);

<CODE ENDS>
```

A.2. mbedTLS

```
<CODE BEGINS>
mbedtls_x509_crt cert;
mbedtls_x509_crt caCert;
uint32_t          certVerifyResultFlags;
...
int result = mbedtls_x509_crt_verify(&cert, &caCert, NULL, NULL,
                                     &certVerifyResultFlags, NULL, NULL);
<CODE ENDS>
```

Appendix B. Constrained BRSKI-EST Message Examples

This appendix extends the message examples from Appendix A of [I-D.ietf-ace-coap-est] with constrained BRSKI messages. The CoAP headers are only fully worked out for the first example, enrollstatus.

B.1. enrollstatus

A coaps enrollstatus message from Pledge to Registrar can be as follows:

```
POST coaps://192.0.2.1:8085/b/es
Content-Format: 60
Payload: <binary CBOR enrollstatus document>
```

The corresponding CoAP header fields are shown below.

```
Ver = 1
T = 0 (CON)
TKL = 1
Code = 0x02 (0.02 is POST method)
Message ID = 0xab0f
Token = 0x4d
Options
  Option (Uri-Path)
    Option Delta = 0xb (option nr = 11)
    Option Length = 0x1
    Option Value = "b"
  Option (Uri-Path)
    Option Delta = 0x0 (option nr = 11)
    Option Length = 0x2
    Option Value = "es"
  Option (Content-Format)
    Option Delta = 0x1 (option nr = 12)
    Option Length = 0x1
    Option Value = 60 (application/cbor)
Payload Marker = 0xFF
Payload = A26776657273696F6E0166737461747573F5 (18 bytes binary)
```

The Uri-Host and Uri-Port Options are omitted because they coincide with the transport protocol (UDP) destination address and port respectively.

The above binary CBOR enrollstatus payload looks as follows in CBOR diagnostic notation, for the case of enrollment success:

```
{
  "version": 1,
  "status": true
}
```

Alternatively the payload could look as follows in case of enrollment failure, using the reason field to describe the failure:

```
Payload = A36776657273696F6E0166737461747573F466726561736F6E782A3C
          496E666F726D61746976652068756D616E207265616461626C652065
          72726F72206D6573736167653E
```

```
{
  "version": 1,
  "status": false,
  "reason": "<Informative human readable error message>"
}
```

To indicate successful reception of the enrollmentstatus telemetry report, a response from the Registrar may then be:

2.04 Changed

With CoAP fields:

```
Ver=1
T=2 (ACK)
TKL=1
Code = 0x44 (2.04 Changed)
Message ID = 0xab0f
Token = 0x4d
```

B.2. voucher_status

A coaps voucher_status message from Pledge to Registrar can be as follows:

```
POST coaps://[2001:db8::2:1]/.well-known/brski/vs
Content-Format: 60 (application/cbor)
Payload:
a46776657273696f6e0166737461747573f466726561736f6e7828496e66
6f726d61746976652068756d616e2d7265616461626c65206572726f7220
6d6573736167656e726561736f6e2d636f6e74657874a100764164646974
696f6e616c20696e666f726d6174696f6e
```

The request payload above is binary CBOR but represented here in hexadecimal for readability. Below is the equivalent CBOR diagnostic format.

```
{
  "version": 1,
  "status": false,
  "reason": "Informative human-readable error message",
  "reason-context": { 0: "Additional information" }
}
```

A success response without payload will then be sent by the Registrar back to the Pledge to indicate reception of the telemetry report:

2.04 Changed

Appendix C. COSE-signed Voucher (Request) Examples

This appendix provides examples of COSE-signed voucher requests and vouchers. First, the used test keys and certificates are described, following by examples of a constrained PVR, RVR and voucher.

C.1. Pledge, Registrar and MASA Keys

This section documents the public and private keys used for all examples in this appendix. These keys are not used in any production system, and must only be used for testing purposes.

C.1.1. Pledge IDevID private key

```
<CODE BEGINS>
Private-Key: (256 bit)
priv:
  9b:4d:43:b6:a9:e1:7c:04:93:45:c3:13:d9:b5:f0:
  41:a9:6a:9c:45:79:73:b8:62:f1:77:03:3a:fc:c2:
  9c:9a
pub:
  04:d6:b7:6f:74:88:bd:80:ae:5f:28:41:2c:72:02:
  ef:5f:98:b4:81:e1:d9:10:4c:f8:1b:66:d4:3e:5c:
  ea:da:73:e6:a8:38:a9:f1:35:11:85:b6:cd:e2:04:
  10:be:fe:d5:0b:3b:14:69:2e:e1:b0:6a:bc:55:40:
  60:eb:95:5c:54
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>
```

C.1.2. Registrar private key

```

<CODE BEGINS>
Private-Key: (256 bit)
priv:
  81:df:bb:50:a3:45:58:06:b5:56:3b:46:de:f3:e9:
  e9:00:ae:98:13:9e:2f:36:68:81:fc:d9:65:24:fb:
  21:7e
pub:
  04:50:7a:c8:49:1a:8c:69:c7:b5:c3:1d:03:09:ed:
  35:ba:13:f5:88:4c:e6:2b:88:cf:30:18:15:4f:a0:
  59:b0:20:ec:6b:eb:b9:4e:02:b8:93:40:21:89:8d:
  a7:89:c7:11:ce:a7:13:39:f5:0e:34:8e:df:0d:92:
  3e:d0:2d:c7:b7
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>

```

C.1.3. MASA private key

```

<CODE BEGINS>
Private-Key: (256 bit)
priv:
  c6:bb:a5:8f:b6:d3:c4:75:28:d8:d3:d9:46:c3:31:
  83:6d:00:0a:9a:38:ce:22:5c:e9:d9:ea:3b:98:32:
  ec:31
pub:
  04:59:80:94:66:14:94:20:30:3c:66:08:85:55:86:
  db:e7:d4:d1:d7:7a:d2:a3:1a:0c:73:6b:01:0d:02:
  12:15:d6:1f:f3:6e:c8:d4:84:60:43:3b:21:c5:83:
  80:1e:fc:e2:37:85:77:97:94:d4:aa:34:b5:b6:c6:
  ed:f3:17:5c:f1
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>

```

C.2. Pledge, Registrar and MASA Certificates

All keys and certificates used for the examples have been generated with OpenSSL – see Appendix D for more details on certificate generation. Below the certificates are listed that accompany the keys shown above. Each certificate description is followed by the hexadecimal representation of the X.509 ASN.1 DER encoded certificate. This representation can be for example decoded using an online ASN.1 decoder.

C.2.1. Pledge IDevID Certificate

<CODE BEGINS>

Certificate:

Data:

Version: 3 (0x2)
Serial Number: 4822678189204992 (0x11223344556600)
Signature Algorithm: ecdsa-with-SHA256
Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=manufacturer,
CN=masa.stok.nl
Validity
Not Before: Dec 9 10:02:36 2020 GMT
Not After : Dec 31 23:59:59 9999 GMT
Subject: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=manufacturing,
CN=uuid:pledge.1.2.3.4/serialNumber=pledge.1.2.3.4
Subject Public Key Info:
Public Key Algorithm: id-ecPublicKey
Public-Key: (256 bit)
pub:
04:d6:b7:6f:74:88:bd:80:ae:5f:28:41:2c:72:02:
ef:5f:98:b4:81:e1:d9:10:4c:f8:1b:66:d4:3e:5c:
ea:da:73:e6:a8:38:a9:f1:35:11:85:b6:cd:e2:04:
10:be:fe:d5:0b:3b:14:69:2e:e1:b0:6a:bc:55:40:
60:eb:95:5c:54
ASN1 OID: prime256v1
NIST CURVE: P-256
X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
X509v3 Authority Key Identifier:
keyid:
E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3

Signature Algorithm: ecdsa-with-SHA256
30:46:02:21:00:d2:e6:45:3b:b0:c3:00:b3:25:8d:f1:83:fe:
d9:37:c1:a2:49:65:69:7f:6b:b9:ef:2c:05:07:06:31:ac:17:
bd:02:21:00:e2:ce:9e:7b:7f:74:50:33:ad:9e:ff:12:4e:e9:
a6:f3:b8:36:65:ab:7d:80:bb:56:88:bc:03:1d:e5:1e:31:6f

<CODE ENDS>

Below is the hexadecimal representation:

<CODE BEGINS>

```
30820226308201cba003020102020711223344556600300a06082a8648ce3d04
0302306f310b3009060355040613024e4c310b300906035504080c024e423110
300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76616e6465
7273746f6b31153013060355040b0c0c6d616e756666163747572657231153013
06035504030c0c6d6173612e73746f6b2e6e6c3020170d323031323039313030
3233365a180f393939393132333313233353935395a308190310b300906035504
0613024e4c310b300906035504080c024e423110300e06035504070c0748656c
6d6f6e6431133011060355040a0c0a76616e64657273746f6b31163014060355
040b0c0d6d616e756666163747572696e67311c301a06035504030c1375756964
3a706c656467652e312e322e332e34311730150603550405130e706c65646765
2e312e322e332e343059301306072a8648ce3d020106082a8648ce3d03010703
420004d6b76f7488bd80ae5f28412c7202ef5f98b481e1d9104cf81b66d43e5c
eada73e6a838a9f1351185b6cde20410befed50b3b14692ee1b06abc554060eb
955c54a32e302c30090603551d1304023000301f0603551d23041830168014e4
0393b4c3d3f42a80a47718f6964903011768a3300a06082a8648ce3d04030203
49003046022100d2e6453bb0c300b3258df183fed937c1a24965697f6bb9ef2c
05070631ac17bd022100e2ce9e7b7f745033ad9eff124ee9a6f3b83665ab7d80
bb5688bc031de51e316f
```

<CODE ENDS>

C.2.2. Registrar Certificate

<CODE BEGINS>

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

70:56:ea:aa:30:66:d8:82:6a:55:5b:90:88:d4:62:bf:9c:f2:8c:fd

Signature Algorithm: ecdsa-with-SHA256

Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=consultancy,
CN=registrar.stok.nl

Validity

Not Before: Dec 9 10:02:36 2020 GMT

Not After : Dec 9 10:02:36 2021 GMT

Subject: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=consultancy,
CN=registrar.stok.nl

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:50:7a:c8:49:1a:8c:69:c7:b5:c3:1d:03:09:ed:
35:ba:13:f5:88:4c:e6:2b:88:cf:30:18:15:4f:a0:
59:b0:20:ec:6b:eb:b9:4e:02:b8:93:40:21:89:8d:
a7:89:c7:11:ce:a7:13:39:f5:0e:34:8e:df:0d:92:
3e:d0:2d:c7:b7

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Subject Key Identifier:

08:C2:BF:36:88:7F:79:41:21:85:87:2F:16:A7:AC:A6:EF:B3:D2:B3

X509v3 Authority Key Identifier:

keyid:

08:C2:BF:36:88:7F:79:41:21:85:87:2F:16:A7:AC:A6:EF:B3:D2:B3

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Extended Key Usage:

CMC Registration Authority, TLS Web Server

Authentication, TLS Web Client Authentication

X509v3 Key Usage: critical

Digital Signature, Non Repudiation, Key Encipherment,

Data Encipherment, Certificate Sign, CRL Sign

Signature Algorithm: ecdsa-with-SHA256

30:44:02:20:74:4c:99:00:85:13:b2:f1:bc:fd:f9:02:1a:46:

fb:17:4c:f8:83:a2:7c:a1:d9:3f:ae:ac:f3:1e:4e:dd:12:c6:

02:20:11:47:14:db:f5:1a:5e:78:f5:81:b9:42:1c:6e:47:02:

ab:53:72:70:c5:ba:fb:2d:16:c3:de:9a:a1:82:c3:5f

<CODE ENDS>

Below is the hexadecimal representation which is in (request-)voucher examples referred to as regis-cert-hex:

```
<CODE BEGINS>
308202753082021ca00302010202147056eaaa3066d8826a555b9088d462bf9c
f28cfd300a06082a8648ce3d0403023073310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31143012060355040b0c0b636f6e
73756c74616e6379311a301806035504030c117265676973747261722e73746f
6b2e6e6c301e170d3230313230393130303233365a170d323131323039313030
3233365a3073310b3009060355040613024e4c310b300906035504080c024e42
3110300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76616e
64657273746f6b31143012060355040b0c0b636f6e73756c74616e6379311a30
1806035504030c117265676973747261722e73746f6b2e6e6c3059301306072a
8648ce3d020106082a8648ce3d03010703420004507ac8491a8c69c7b5c31d03
09ed35ba13f5884ce62b88cf3018154fa059b020ec6bebb94e02b8934021898d
a789c711cea71339f50e348edf0d923ed02dc7b7a3818d30818a301d0603551d
0e0416041408c2bf36887f79412185872f16a7aca6efb3d2b3301f0603551d23
04183016801408c2bf36887f79412185872f16a7aca6efb3d2b3300f0603551d
130101fff040530030101fff30270603551d250420301e06082b0601050507031c
06082b0601050507030106082b06010505070302300e0603551d0f0101fff0404
030201f6300a06082a8648ce3d04030203470030440220744c99008513b2f1bc
fdf9021a46fb174cf883a27cald93faeacf31e4edd12c60220114714dbf51a5e
78f581b9421c6e4702ab537270c5bafb2dl6c3de9aa182c35f
<CODE ENDS>
```

C.2.3. MASA Certificate

```
<CODE BEGINS>
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      14:26:b8:1c:ce:d8:c3:e8:14:05:cb:87:67:0d:be:ef:d5:81:25:b4
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok,
      OU=manufacturer, CN=masa.stok.nl

    Validity
      Not Before: Dec  9 10:02:36 2020 GMT
      Not After : Sep  5 10:02:36 2023 GMT
    Subject: C=NL, ST=NB, L=Helmond, O=vanderstok,
      OU=manufacturer, CN=masa.stok.nl
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:59:80:94:66:14:94:20:30:3c:66:08:85:55:86:
```

```
db:e7:d4:d1:d7:7a:d2:a3:1a:0c:73:6b:01:0d:02:
12:15:d6:1f:f3:6e:c8:d4:84:60:43:3b:21:c5:83:
80:1e:fc:e2:37:85:77:97:94:d4:aa:34:b5:b6:c6:
ed:f3:17:5c:f1
ASN1 OID: prime256v1
NIST CURVE: P-256
X509v3 extensions:
  X509v3 Subject Key Identifier:
E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3
  X509v3 Authority Key Identifier:
    keyid:
E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3

  X509v3 Basic Constraints: critical
    CA:TRUE
  X509v3 Extended Key Usage:
    CMC Registration Authority,
    TLS Web Server Authentication,
    TLS Web Client Authentication
  X509v3 Key Usage: critical
    Digital Signature, Non Repudiation, Key Encipherment,
    Data Encipherment, Certificate Sign, CRL Sign
Signature Algorithm: ecdsa-with-SHA256
30:44:02:20:2e:c5:f2:24:72:70:20:ea:6e:74:8b:13:93:67:
8a:e6:fe:fb:8d:56:7f:f5:34:18:a9:ef:a5:0f:c3:99:ca:53:
02:20:3d:dc:91:d0:e9:6a:69:20:01:fb:e4:20:40:de:7c:7d:
98:ed:d8:84:53:61:84:a7:f9:13:06:4c:a9:b2:8f:5c
<CODE ENDS>
```

Below is the hexadecimal representation:

```
<CODE BEGINS>
3082026d30820214a00302010202141426b81cced8c3e81405cb87670dbeefd5
8125b4300a06082a8648ce3d040302306f310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31153013060355040b0c0c6d616e
7566616374757265723115301306035504030c0c6d6173612e73746f6b2e6e6c
301e170d3230313230393130303233365a170d3233303930353130303233365a
306f310b3009060355040613024e4c310b300906035504080c024e423110300e
06035504070c0748656c6d6f6e6431133011060355040a0c0a76616e64657273
746f6b31153013060355040b0c0c6d616e756661637475726572311530130603
5504030c0c6d6173612e73746f6b2e6e6c3059301306072a8648ce3d02010608
2a8648ce3d0301070342000459809466149420303c6608855586dbe7d4d1d77a
d2a31a0c736b010d021215d61ff36ec8d48460433b21c583801efce237857797
94d4aa34b5b6c6edf3175cf1a3818d30818a301d0603551d0e04160414e40393
b4c3d3f42a80a47718f6964903011768a3301f0603551d23041830168014e403
93b4c3d3f42a80a47718f6964903011768a3300f0603551d130101ff04053003
0101ff30270603551d250420301e06082b0601050507031c06082b0601050507
030106082b06010505070302300e0603551d0f0101ff0404030201f6300a0608
2a8648ce3d040302034700304402202ec5f224727020ea6e748b1393678ae6fe
fb8d567ff53418a9efa50fc399ca5302203ddc91d0e96a692001fbe42040de7c
7d98edd884536184a7f913064ca9b28f5c
<CODE ENDS>
```

C.3. COSE-signed Pledge Voucher Request (PVR)

In this COSE example the voucher request has been signed by the Pledge using the private key of Appendix C.1.1, and has been sent to the link-local JRC (Registrar) over CoAPS.

```
POST coaps://[JRC-link-local-address]/b/rv
Content-Format: TBD3
Payload: signed_request_voucher
```

The payload `signed_request_voucher` is shown as hexadecimal dump (with lf added):

```
<CODE BEGINS>
d28444a101382ea104582097113db094eee8eae48683e7337875c0372164
be89d023a5f3df52699c0fbfb55902d2a11909c5a60274323032302d3132
2d32335431323a30353a32325a0474323032322d31322d32335431323a30
353a32325a01020750684ca83e27230aff97630cf2c1ec409a0d6e706c65
6467652e312e322e332e340a590279308202753082021ca0030201020214
7056eaaa3066d8826a555b9088d462bf9cf28cfd300a06082a8648ce3d04
03023073310b3009060355040613024e4c310b300906035504080c024e42
3110300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76
616e64657273746f6b31143012060355040b0c0b636f6e73756c74616e63
79311a301806035504030c117265676973747261722e73746f6b2e6e6c30
1e170d3230313230393130303233365a170d323131323039313030323336
5a3073310b3009060355040613024e4c310b300906035504080c024e4231
10300e06035504070c0748656c6d6f6e6431133011060355040a0c0a7661
6e64657273746f6b31143012060355040b0c0b636f6e73756c74616e6379
311a301806035504030c117265676973747261722e73746f6b2e6e6c3059
301306072a8648ce3d020106082a8648ce3d03010703420004507ac8491a
8c69c7b5c31d0309ed35ba13f5884ce62b88cf3018154fa059b020ec6beb
b94e02b8934021898da789c711cea71339f50e348edf0d923ed02dc7b7a3
818d30818a301d0603551d0e0416041408c2bf36887f79412185872f16a7
aca6efb3d2b3301f0603551d2304183016801408c2bf36887f7941218587
2f16a7aca6efb3d2b3300f0603551d130101ff040530030101ff30270603
551d250420301e06082b0601050507031c06082b0601050507030106082b
06010505070302300e0603551d0f0101ff0404030201f6300a06082a8648
ce3d04030203470030440220744c99008513b2f1bcfd9021a46fb174cf8
83a27ca1d93faeacf31e4edd12c60220114714dbf51a5e78f581b9421c6e
4702ab537270c5bafb2d16c3de9aa182c35f58473045022063766c7bbd1b
339dbc398e764af3563e93b25a69104befe9aac2b3336b8f56e1022100cd
0419559ad960ccaed4dee3f436eca40b7570b25a52eb60332bc1f2991484
e9
<CODE ENDS>
```

The Pledge uses the "proximity" (SID 2502, enum 2) assertion together with an included proximity-registrar-cert field (SID 2511) to inform MASA about its proximity to the specific Registrar. The representation of signed_voucher_request in CBOR diagnostic format is:

```

<CODE BEGINS>
Diagnose(signed_request_voucher) =
18([
h'A101382E',      / {"alg": -47} /
{4: h' 97113DB094EEE8EAE48683E7337875C0372164B
    E89D023A5F3DF52699C0FBFB5'},
h'<request_voucher>', / byte string as detailed below /
h' 3045022063766C7BBD1B339DBC398E764AF3563E93B
25A69104BEFE9AAC2B3336B8F56E1022100CD0419559A
D960CCAED4DEE3F436ECA40B7570B25A52EB60332BC1F
2991484E9'
])

Diagnose(request_voucher) =
{2501: {2: "2020-12-23T12:05:22Z",
        4: "2022-12-23T12:05:22Z",
        1: 2,
        7: h' 684CA83E27230AFF97630CF2C1EC409A',
        13: "pledge.1.2.3.4",
        10: h'<regis-cert-hex>' / byte string as defined in C.2.2 /
      }}
<CODE ENDS>

```

C.4. COSE-signed Registrar Voucher Request (RVR)

In this example the Registrar's voucher request has been signed by the JRC (Registrar) using the private key from Appendix C.1.2. Contained within this voucher request is the voucher request PVR that was made by the Pledge to JRC. Note that the RVR uses the HTTPS protocol (not CoAP) and corresponding long URI path names as defined in [RFC8995]. The Content-Type and Accept headers indicate the constrained voucher format that is defined in the present document. Because the Pledge used this format in the PVR, the JRC must also use this format in the RVR.

```

POST https://masa.example.com/.well-known/brski/requestvoucher
Content-Type: application/voucher-cose+cbor
Accept: application/voucher-cose+cbor
Body: signed_masa_request_voucher

```

The payload `signed_masa_voucher_request` is shown as hexadecimal dump (with lf added):

<CODE BEGINS>

```
d28444a101382ea1045820e8735bc4b470c3aa6a7aa9aa8ee584c09c1113
1b205efec5d0313bad84c5cd05590414a11909c5a60274323032302d3132
2d32385431303a30333a33355a0474323032322d31322d32385431303a30
333a33355a07501551631f6e0416bd162ba53ea00c2a050d6e706c656467
652e312e322e332e3405587131322d32385431303a30333a33355a075015
51631f6e0416bd162ba53ea00c2a050d6e706c656467652e312e322e332e
3405587131322d32385431303a300000000000000000000000000000416bd16
2ba53ea00c2a050d6e706c656467652e312e322e332e3405587131322d32
385431303a09590349d28444a101382ea104582097113db094eee8eae486
83e7337875c0372164be89d023a5f3df52699c0fbfb55902d2a11909c5a6
0274323032302d31322d32385431303a30333a33355a0474323032322d31
322d32385431303a30333a33355a010207501551631f6e0416bd162ba53e
a00c2a050d6e706c656467652e312e322e332e340a590279308202753082
021ca00302010202147056eaaa3066d8826a555b9088d462bf9cf28cfd30
0a06082a8648ce3d0403023073310b3009060355040613024e4c310b3009
06035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31143012060355040b0c0b63
6f6e73756c74616e6379311a301806035504030c11726567697374726172
2e73746f6b2e6e6c301e170d3230313230393130303233365a170d323131
3230393130303233365a3073310b3009060355040613024e4c310b300906
035504080c024e423110300e06035504070c0748656c6d6f6e6431133011
060355040a0c0a76616e64657273746f6b31143012060355040b0c0b636f
6e73756c74616e6379311a301806035504030c117265676973747261722e
73746f6b2e6e6c3059301306072a8648ce3d020106082a8648ce3d030107
03420004507ac8491a8c69c7b5c31d0309ed35ba13f5884ce62b88cf3018
154fa059b020ec6bebb94e02b8934021898da789c711cea71339f50e348e
df0d923ed02dc7b7a3818d30818a301d0603551d0e0416041408c2bf3688
7f79412185872f16a7aca6efb3d2b3301f0603551d2304183016801408c2
bf36887f79412185872f16a7aca6efb3d2b3300f0603551d130101ff0405
30030101ff30270603551d250420301e06082b0601050507031c06082b06
01050507030106082b06010505070302300e0603551d0f0101ff04040302
01f6300a06082a8648ce3d04030203470030440220744c99008513b2f1bc
fdf9021a46fb174cf883a27cald93faeacf31e4edd12c60220114714dbf5
1a5e78f581b9421c6e4702ab537270c5bafb2d16c3de9aa182c35f584730
45022063766c7bbdlb339dbc398e764af3563e93b25a69104befe9aac2b3
336b8f56e1022100cd0419559ad960ccaed4dee3f436eca40b7570b25a52
eb60332bc1f2991484e958473045022100e6b45558c1b806bba23f4ac626
c9bdb6fd354ef4330d8dfb7c529f29cca934c802203c1f2ccbbac89733d1
7ee7775bc2654c5f1cc96afba2741cc31532444aa8fed8
```

<CODE ENDS>

The representation of signed_masa_voucher_request in CBOR diagnostic format is:

```

<CODE BEGINS>
Diagnose(signed_registrar_request-voucher)
18([
h'A101382E',      / {"alg": -47} /
h'E8735BC4B470C3AA6A7AA9AA8EE584C09C11131B205EFEC5D0313BAD84
C5CD05'},
h'<registrar_request_voucher>', / byte string as detailed below /
h'3045022100E6B45558C1B806BBA23F4AC626C9BDB6FD354EF4330D8DFB
7C529F29CCA934C802203C1F2CCBBAC89733D17EE7775BC2654C5F1CC96A
FBA2741CC31532444AA8FED8'
])

Diagnose(registrar_request_voucher)
{2501:
  {2: "2020-12-28T10:03:35Z",
   4: "2022-12-28T10:03:35Z",
   7: h'1551631F6E0416BD162BA53EA00C2A05',
  13: "pledge.1.2.3.4",
   5: h'31322D32385431303A30333A333355A07501551631F6E0416BD
      162BA53EA00C2A050D6E706C656467652E312E322E332E3405
      587131322D32385431303A30000000000000000000000000000004
      16BD162BA53EA00C2A050D6E706C656467652E312E322E332E
      3405587131322D32385431303A', / idevid-issuer /
   9: h'<prior-pvr>' / prior-signed-voucher-request = PVR /
  }
}
<CODE ENDS>

```

C.5. COSE-signed Voucher from MASA

The resulting voucher is created by the MASA and returned via the JRC to the Pledge. It is signed by the MASA's private key (see Appendix C.1.3) and can be verified by the Pledge using the MASA's public key that it stores.

Below is the binary signed_voucher, encoded in hexadecimal (with lf added):

```

<CODE BEGINS>
d28444a101382ea104582039920a34ee92d3148ab3a729f58611193270c9
029f7784daf112614b19445d5158cfa1190993a70274323032302d31322d
32335431353a30333a31325a0474323032302d31322d32335431353a3233
3a31325a010007506508e06b2959d5089d7a3169ea889a490b6e706c6564
67652e312e322e332e340858753073310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e6431
133011060355040a0c0a76616e64657273746f6b31143012060355040b0c
0b636f6e73756c74616e6379311a301806035504030c1172656769737472
61722e73746f6b2e6e6c03f458473045022022515d96cd12224ee5d3ac67
3237163bba24ad84815699285d9618f463ee73fa022100a6bff9d8585c1c
9256371ece94da3d26264a5dfec0a354fe7b3aef58344c512f
<CODE ENDS>

```

The representation of signed_voucher in CBOR diagnostic format is:

```

<CODE BEGINS>
Diagnose(signed_voucher) =
18([
h'A101382E',      / {"alg": -47} /
{4: h'39920A34EE92D3148AB3A729F58611193270C9029F7784DAF112614B194
45D51'},
h'<voucher>',      / byte string as detailed below /
h'3045022022515D96CD12224EE5D3AC673237163BBA24AD84815699285D9618F
463EE73FA022100A6BFF9D8585C1C9256371ECE94DA3D26264A5DFEC0A354FE7B
3AEF58344C512F'
])

Diagnose(voucher) =
{2451:
  {2: "2020-12-23T15:03:12Z",
   4: "2020-12-23T15:23:12Z",
   1: 0,
   7: h'6508E06B2959D5089D7A3169EA889A49',
  11: "pledge.1.2.3.4",
   8: h'<regis-cert-hex>', / as detailed in C.2.2 /
   3: false}
}
<CODE ENDS>

```

In above, regis-cert-hex represents the hexadecimal encoding of the Registrar certificate of Appendix C.2.2.

Appendix D. Generating Certificates with OpenSSL

This informative appendix shows an example of a Bash shell script to generate test certificates for the Pledge IDevID, the Registrar and the MASA. This shell script cannot be run stand-alone because it depends on particular input files which are not included in this appendix. Nevertheless, this example script may provide guidance on how OpenSSL can be configured for generating Constrained BRSKI certificates.

Note: the *-comb.crt certificate files combine the certificate with the private key. These are generated to be used by libcoap for DTLS connection establishment.

```
<CODE BEGINS>
#!/bin/bash
#try-cert.sh
export dir=./brski/intermediate
export cadir=./brski
export cnfdir=./conf
export format=pem
export default_crl_days=30
sn=8

DevID=pledge.1.2.3.4
serialNumber="serialNumber=$DevID"
export hwType=1.3.6.1.4.1.6715.10.1
export hwSerialNum=01020304 # Some hex
export subjectAltName="otherName:1.3.6.1.5.5.7.8.4;SEQ:hmodname"
echo $hwType - $hwSerialNum
echo $serialNumber
OPENSSL_BIN="openssl"

# remove all files
rm -r ./brski/*
#
# initialize file structure
# root level
cd $cadir
mkdir certs crl csr newcerts private
chmod 700 private
touch index.txt
touch serial
echo 11223344556600 >serial
echo 1000 >crlnumber
# intermediate level
mkdir intermediate
cd intermediate
```

```
mkdir certs crl csr newcerts private
chmod 700 private
touch index.txt
echo 11223344556600 >serial
echo 1000 > crlnumber
cd ../..

# file structure is cleaned start filling

echo "#####"
echo "create registrar keys and certificates "
echo "#####"

echo "create root registrar certificate using ecdsa with sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey \
    -noout -out $cadir/private/ca-regis.key

$OPENSSL_BIN req -new -x509 \
    -config $cnfdir/openssl-regis.cnf \
    -key $cadir/private/ca-regis.key \
    -out $cadir/certs/ca-regis.crt \
    -extensions v3_ca \
    -days 365 \
    -subj "/C=NL/ST=NB/L=Helmond/O=vanderstok/OU=consultancy \
/CN=registrar.stok.nl"

# Combine authority certificate and key
echo "Combine authority certificate and key"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
    -inkey $cadir/private/ca-regis.key \
    -in $cadir/certs/ca-regis.crt -export \
    -out $cadir/certs/ca-regis-comb.pfx

# converteer authority pkcs12 file to pem
echo "converteer authority pkcs12 file to pem"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
    -in $cadir/certs/ca-regis-comb.pfx \
    -out $cadir/certs/ca-regis-comb.crt -nodes

#show certificate in registrar combined certificate
$OPENSSL_BIN x509 -in $cadir/certs/ca-regis-comb.crt -text

#
# Certificate Authority for MASA
#
```

```
echo "#####"
echo "create MASA keys and certificates "
echo "#####"

echo "create root MASA certificate using ecdsa with sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey -noout \
    -out $cadir/private/ca-masa.key

$OPENSSL_BIN req -new -x509 \
    -config $cnfdir/openssl-masa.cnf \
    -days 1000 -key $cadir/private/ca-masa.key \
    -out $cadir/certs/ca-masa.crt \
    -extensions v3_ca \
    -subj "/C=NL/ST=NB/L=Helmond/O=vanderstok/OU=manufacturer\
/CN=masa.stok.nl"

# Combine authority certificate and key
echo "Combine authority certificate and key for masa"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
    -inkey $cadir/private/ca-masa.key \
    -in $cadir/certs/ca-masa.crt -export \
    -out $cadir/certs/ca-masa-comb.pfx

# converteer authority pkcs12 file to pem for masa
echo "converteer authority pkcs12 file to pem for masa"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
    -in $cadir/certs/ca-masa-comb.pfx \
    -out $cadir/certs/ca-masa-comb.crt -nodes

#show certificate in pledge combined certificate
$OPENSSL_BIN x509 -in $cadir/certs/ca-masa-comb.crt -text

#
# Certificate for Pledge derived from MASA certificate
#
echo "#####"
echo "create pledge keys and certificates "
echo "#####"

# Pledge derived Certificate

echo "create pledge derived certificate using ecdsa with sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey -noout \
    -out $dir/private/pledge.key

echo "create pledge certificate request"
```

```
$OPENSSL_BIN req -nodes -new -sha256 \  
-key $dir/private/pledge.key -out $dir/csr/pledge.csr \  
-subj "/C=NL/ST=NB/L=Helmond/O=vanderstok/OU=manufacturing\  
/CN=uuid:$DevID/$serialNumber"  
  
# Sign pledge derived Certificate  
echo "sign pledge derived certificate "  
$OPENSSL_BIN ca -config $cnfdir/openssl-pledge.cnf \  
-extensions 8021ar_idevid\  
-days 365 -in $dir/csr/pledge.csr \  
-out $dir/certs/pledge.crt  
  
# Add pledge key and pledge certificate to pkcs12 file  
echo "Add derived pledge key and derived pledge \  
certificate to pkcs12 file"  
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\  
-inkey $dir/private/pledge.key \  
-in $dir/certs/pledge.crt -export \  
-out $dir/certs/pledge-comb.pfx  
  
# converteer pledge pkcs12 file to pem  
echo "converteer pledge pkcs12 file to pem"  
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\  
-in $dir/certs/pledge-comb.pfx \  
-out $dir/certs/pledge-comb.crt -nodes  
  
#show certificate in pledge-comb.crt  
$OPENSSL_BIN x509 -in $dir/certs/pledge-comb.crt -text  
  
#show private key in pledge-comb.crt  
$OPENSSL_BIN ecparam -name prime256v1\  
-in $dir/certs/pledge-comb.crt -text  
  
<CODE ENDS>
```

Appendix E. Pledge Device Class Profiles

This specification allows implementers to select between various functional options for the Pledge, yielding different code size footprints and different requirements on Pledge hardware. Thus for each product an optimal trade-off between functionality, development/maintenance cost and hardware cost can be made.

This appendix illustrates different selection outcomes by means of defining different example "profiles" of constrained Pledges. In the following subsections, these profiles are defined and a comparison is provided.

E.1. Minimal Pledge

The Minimal Pledge profile (Min) aims to reduce code size and hardware cost to a minimum. This comes with some severe functional restrictions, in particular:

- * No support for EST re-enrollment: whenever this would be needed, a factory reset followed by a new bootstrap process is required.
- * No support for change of Registrar: for this case, a factory reset followed by a new bootstrap process is required.

This profile would be appropriate for single-use devices which must be replaced rather than re-deployed. That might include medical devices, but also sensors used during construction, such as concrete temperature sensors.

E.2. Typical Pledge

The Typical Pledge profile (Typ) aims to support a typical Constrained BRSKI feature set including EST re-enrollment support and Registrar changes.

E.3. Full-featured Pledge

The Full-featured Pledge profile (Full) illustrates a Pledge category that supports multiple bootstrap methods, hardware real-time clock, BRSKI/EST resource discovery, and CSR Attributes request/response. It also supports most of the optional features defined in this specification.

E.4. Comparison Chart of Pledge Classes

The below table specifies the functions implemented in the three example Pledge classes Min, Typ and Full.

Function	Min	Typ	Full
General	===	===	====
Support Constrained BRSKI bootstrap	Y	Y	Y
Support other bootstrap method(s)	-	-	Y
Real-time clock and cert time checks	-	-	Y
Constrained BRSKI	===	===	====

Discovery for rt=brski*	-	-	Y
Support pinned Registrar public key (RPK)	Y	-	Y
Support pinned Registrar certificate	-	Y	Y
Support pinned Domain CA	-	Y	Y
Constrained EST	===	===	=====
Discovery for rt=ace.est*	-	-	Y
GET /att and response parsing	-	-	Y
GET /crtts format 281 (multiple CA certs)	-	-	Y
GET /crtts only format TBD287 (one CA cert only)	Y	Y	-
ETag handling support for GET /crtts	-	Y	Y
Re-enrollment supported	- (1)	Y	Y
6.6.1 optimized procedure	Y	Y	-
Pro-active cert re-enrollment at own initiative	N/A	-	Y
Periodic trust anchor retrieval GET /crtts	- (1)	Y	Y
Supports change of Registrar identity	- (1)	Y	Y

Table 6

Notes: (1) is possible only by doing a factory-reset followed by a new bootstrap procedure.

Contributors

Russ Housley
Email: housley@vigilsec.com

Authors' Addresses

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

Peter van der Stok
vanderstok consultancy
Email: stokcons@bbhmail.nl

Panos Kampanakis
Cisco Systems
Email: pkampana@cisco.com

Esko Dijk
IoTconsultancy.nl
Email: esko.dijk@iotconsultancy.nl

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 8 July 2021

B. E. Carpenter
Univ. of Auckland
B. Liu, Ed.
Huawei Technologies
W. Wang
X. Gong
BUPT University
4 January 2021

Generic Autonomic Signaling Protocol Application Program Interface
(GRASP API)
draft-ietf-anima-grasp-api-10

Abstract

This document is a conceptual outline of an application programming interface (API) for the Generic Autonomic Signaling Protocol (GRASP). Such an API is needed for Autonomic Service Agents (ASA) calling the GRASP protocol module to exchange autonomic network messages with other ASAs. Since GRASP is designed to support asynchronous operations, the API will need to be adapted according to the support for asynchronicity in various programming languages and operating systems.

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 8 July 2021.

Copyright Notice

Copyright (c) 2021 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	2
2. GRASP API for ASA	5
2.1. Design Assumptions	5
2.2. Asynchronous Operations	6
2.2.1. Alternative Asynchronous Mechanisms	6
2.2.2. Multiple Negotiation Scenario	7
2.2.3. Overlapping Sessions and Operations	8
2.2.4. Session Termination	9
2.3. API definition	9
2.3.1. Overview of Functions	9
2.3.2. Parameters and data structures	10
2.3.3. Registration	15
2.3.4. Discovery	17
2.3.5. Negotiation	19
2.3.6. Synchronization and Flooding	26
2.3.7. Invalid Message Function	31
3. Implementation Status [RFC Editor: please remove]	31
4. Security Considerations	31
5. IANA Considerations	32
6. Acknowledgements	33
7. References	33
7.1. Normative References	33
7.2. Informative References	33
Appendix A. Error Codes	34
Appendix B. Change log [RFC Editor: Please remove]	36
Authors' Addresses	39

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. These nodes are members of a secure Autonomic Control Plane (ACP) such as defined by [I-D.ietf-anima-autonomic-control-plane].

When ASAs communicate with each other, they should use the Generic Autonomic Signaling Protocol (GRASP) [I-D.ietf-anima-grasp]. GRASP relies on the message confidentiality and integrity provided by the ACP, with the consequence that all nodes in a given autonomic network share the same trust boundary, i.e., the boundary of the ACP. Nodes that have not successfully joined the ACP cannot send, receive or intercept GRASP messages via the ACP, and cannot usurp ACP addresses. An ASA runs in an ACP node and therefore benefits from the node's security properties when transmitting over the ACP, i.e., message integrity, message confidentiality and the fact that unauthorized nodes cannot join the ACP. All ASAs within a given autonomic network therefore trust each other's messages. For these reasons, the API defined in this document has no explicit security features.

An important feature of GRASP is the concept of a GRASP objective. This is a data structure encoded, like all GRASP messages, in CBOR [RFC8949]. Its main contents are a name and a value, explained at more length in the 'Terminology' section of [I-D.ietf-anima-grasp]. When an objective is passed from one ASA to another using GRASP, its value is either conveyed in one direction (by a process of synchronization or flooding), or negotiated bilaterally. The semantics of the value are opaque to GRASP and therefore to the API. Each objective must be accurately specified in a dedicated specification, as discussed in the 'Objective Options' section of [I-D.ietf-anima-grasp]. In particular, the specification will define the syntax and semantics of the value of the objective, whether and how it supports a negotiation process, whether it supports a dry run mode, and any other details needed for interoperability. The use of CBOR, with CDDL [RFC8610] as the data definition language, allows the value to be passed between ASAs regardless of the programming languages in use. Data storage and consistency during negotiation are the responsibility of the ASAs involved. Additionally, GRASP needs to cache the latest values of objectives that are received by flooding.

As Figure 1 shows, a GRASP implementation could contain several sub-layers. The bottom layer is the GRASP base protocol module, which is only responsible for sending and receiving GRASP messages and maintaining shared data structures. Above that is the basic API described in this document. The upper layer contains some extended API functions based upon GRASP basic protocol. For example, [I-D.ietf-anima-grasp-distribution] describes a possible extended function.

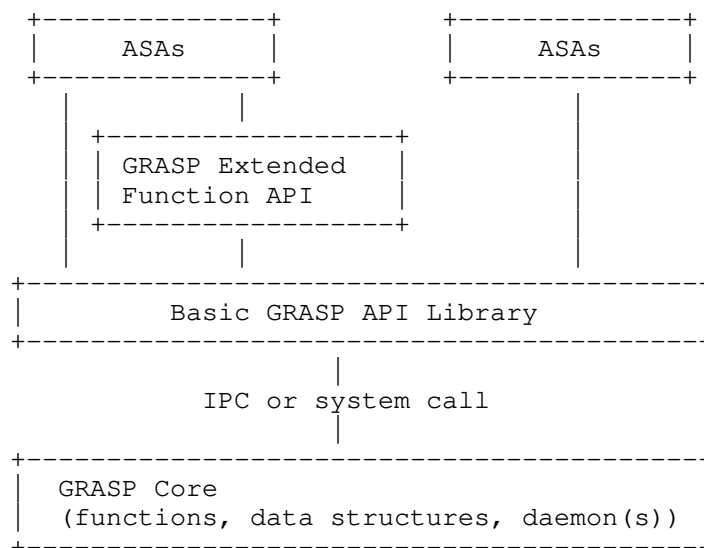


Figure 1: Software layout

Multiple ASAs in a single node will share the same instance of GRASP, much as multiple applications share a single TCP/IP stack. This aspect is hidden from individual ASAs by the API, and is not further discussed here.

It is desirable that ASAs can be designed as portable user-space programs using a system-independent API. In many implementations, the GRASP code will therefore be split between user space and kernel space. In user space, library functions provide the API and communicate directly with ASAs. In kernel space is a daemon, or a set of sub-services, providing GRASP core functions that are independent of specific ASAs, such as multicast handling and relaying, and common data structures such as the discovery cache. The GRASP API library would need to communicate with the GRASP core via an inter-process communication (IPC) or system call mechanism. The details of this are system-dependent.

Both the GRASP library and the extended function modules should be available to the ASAs. However, since the extended functions are expected to be added in an incremental manner, they will be the subject of future documents. This document only describes the basic GRASP API.

The functions provided by the API do not map one-to-one onto GRASP messages. Rather, they are intended to offer convenient support for message sequences (such as a discovery request followed by responses

from several peers, or a negotiation request followed by various possible responses). This choice was made to assist ASA programmers in writing code based on their application requirements rather than needing to understand protocol details.

Note that a simple autonomic node might contain very few ASAs in addition to the autonomic infrastructure components described in [I-D.ietf-anima-bootstrapping-keyinfra] and [I-D.ietf-anima-autonomic-control-plane]. Such a node might directly integrate a GRASP protocol stack in its code and therefore not require this API to be installed. However, the programmer would then need a deeper understanding of the GRASP protocol than is needed to use the API.

This document gives a conceptual outline of the API. It is not a formal specification for any particular programming language or operating system, and it is expected that details will be clarified in individual implementations.

2. GRASP API for ASA

2.1. Design Assumptions

The assumption of this document is that an Autonomic Service Agent (ASA) needs to call a separate GRASP implementation. The latter handles protocol details (security, sending and listening for GRASP messages, waiting, caching discovery results, negotiation looping, sending and receiving synchronization data, etc.) but understands nothing about individual GRASP objectives (Section 2.10 of [I-D.ietf-anima-grasp]). The semantics of objectives are unknown to the GRASP protocol and are handled only by the ASAs. Thus, this is an abstract API for use by ASAs. Individual language bindings should be defined in separate documents.

Different ASAs may make different use of GRASP features, such as:

- * Use GRASP only for discovery purposes.
- * Use GRASP negotiation but only as an initiator (client).
- * Use GRASP negotiation but only as a responder.
- * Use GRASP negotiation as an initiator or responder.
- * Use GRASP synchronization but only as an initiator (recipient).
- * Use GRASP synchronization but only as a responder and/or flooder.

- * Use GRASP synchronization as an initiator, responder and/or flooder.

The API also assumes that one ASA may support multiple objectives. Nothing prevents an ASA from supporting some objectives for synchronization and others for negotiation.

The API design assumes that the operating system and programming language provide a mechanism for simultaneous asynchronous operations. This is discussed in detail in Section 2.2.

A few items are out of scope in this version, since practical experience is required before including them:

- * Authorization of ASAs is not defined as part of GRASP and is a subject for future study.
- * User-supplied explicit locators for an objective are not supported. The GRASP core will supply the locator, using the IP address of the node concerned.
- * The Rapid mode of GRASP (Section 2.5.4 of [I-D.ietf-anima-grasp]) is not supported.

2.2. Asynchronous Operations

GRASP depends on asynchronous operations and wait states, and some of its messages are not idempotent, meaning that repeating a message may cause repeated changes of state in the recipient ASA. Many ASAs will need to support several concurrent operations; for example an ASA might need to negotiate one objective with a peer while discovering and synchronizing a different objective with a different peer. Alternatively, an ASA which acts as a resource manager might need to run simultaneous negotiations for a given objective with multiple different peers. Such an ASA will probably need to support uninterruptible atomic changes to its internal data structures, using a mechanism provided by the operating system and programming language in use.

2.2.1. Alternative Asynchronous Mechanisms

Thus, some ASAs need to support asynchronous operations, and therefore the GRASP core must do so. Depending on both the operating system and the programming language in use, there are various techniques for such parallel operations, three of which we consider here: multi-threading, an event loop structure using polling, and an event loop structure using callback functions.

1. In multi-threading, the operating system and language will provide the necessary support for asynchronous operations, including creation of new threads, context switching between threads, queues, locks, and implicit wait states. In this case, API calls can be treated as simple synchronous function calls within their own thread, even if the function includes wait states, blocking and queueing. Concurrent operations will each run in their own threads. For example, the discover() call may not return until discovery results have arrived or a timeout has occurred. If the ASA has other work to do, the discover() call must be in a thread of its own.
2. In an event loop implementation with polling, blocking calls are not acceptable. Therefore all calls must be non-blocking, and the main loop could support multiple GRASP sessions in parallel by repeatedly polling each one for a change of state. To facilitate this, the API implementation would provide non-blocking versions of all the functions that otherwise involve blocking and queueing. In these calls, a 'noReply' code will be returned by each call instead of blocking, until such time as the event for which it is waiting (or a failure) has occurred. Thus, for example, discover() would return 'noReply' instead of waiting until discovery has succeeded or timed out. The discover() call would be repeated in every cycle of the main loop until it completes. Effectively, it becomes a polling call.
3. It was noted earlier that some GRASP messages are not idempotent; in particular this applies to each step in a negotiation session - sending the same message twice might produce unintended side effects. This is not affected by event loop polling: repeating a call after a 'noReply' does not repeat a message; it simply checks whether a reply has been received.
4. In an event loop implementation with callbacks, the ASA programmer would provide a callback function for each asynchronous operation. This would be called asynchronously when a reply is received or a failure such as a timeout occurs.

2.2.2. Multiple Negotiation Scenario

The design of GRASP allows the following scenario. Consider an ASA "A" that acts as a resource allocator for some objective. An ASA "B" launches a negotiation with "A" to obtain or release a quantity of the resource. While this negotiation is under way, "B" chooses to launch a second simultaneous negotiation with "A" for a different quantity of the same resource. "A" must therefore conduct two separate negotiation sessions at the same time with the same peer, and must not mix them up.

Note that ASAs could be designed to avoid such a scenario, i.e. restricted to exactly one negotiation session at a time for a given objective, but this would be a voluntary restriction not required by the GRASP protocol. In fact it is an assumption of GRASP that any ASA managing a resource may need to conduct multiple parallel negotiations, possibly with the same peer. Communication patterns could be very complex, with a group of ASAs overlapping negotiations among themselves, as described in [I-D.ciavaglia-anima-coordination]. Therefore, the API design allows for such scenarios.

In the callback model, for the scenario just described, the ASAs "A" and "B" will each provide two instances of the callback function, one for each session. For this reason, each ASA must be able to distinguish the two sessions, and the peer's IP address is not sufficient for this. It is also not safe to rely on transport port numbers for this, since future variants of GRASP might use shared ports rather than a separate port per session. Hence the GRASP design includes a session identifier. Thus, when necessary, a session handle (see next section) is used in the API to distinguish simultaneous GRASP sessions from each other, so that any number of sessions may proceed asynchronously in parallel.

2.2.3. Overlapping Sessions and Operations

A GRASP session consists of a finite sequence of messages (for discovery, synchronization, or negotiation) between two ASAs. It is uniquely identified on the wire by a pseudo-random session identifier plus the IP address of the initiator of the session. Further details are given in the section 'Session Identifier' of [I-D.ietf-anima-grasp].

On the first call in a new GRASP session, the API returns a 'session_handle' handle that uniquely identifies the session within the API, so that multiple overlapping sessions can be distinguished. A likely implementation is to form the handle from the underlying GRASP Session ID and IP address. This handle must be used in all subsequent calls for the same session. Also see Section 2.3.2.8.

An additional mechanism that might increase efficiency for polling implementations is to add a general call, say notify(), which would check the status of all outstanding operations for the calling ASA and return the session_handle values for all sessions that have changed state. This would eliminate the need for repeated calls to the individual functions returning a 'noReply'. This call is not described below as the details are likely to be implementation-specific.

An implication of the above for all GRASP implementations is that the GRASP core must keep state for each GRASP operation in progress, most likely keyed by the GRASP Session ID and the GRASP source address of the session initiator. Even in a threaded implementation, the GRASP core will need such state internally. The `session_handle` parameter exposes this aspect of the implementation.

2.2.4. Session Termination

GRASP sessions may terminate for numerous reasons. A session ends when discovery succeeds or times out, when negotiation succeeds or fails, when a synchronization result is delivered, when the other end fails to respond before a timeout expires, when a loop count expires, or when a network socket error occurs. Note that a timeout at one end of a session might result in a timeout or a socket error at the other end, since GRASP does not send error messages in this case. In all cases, the API will return an appropriate code to the caller, which should then release any reserved resources. After failure cases, the GRASP specification recommends an exponential backoff before retrying.

2.3. API definition

2.3.1. Overview of Functions

The functions provided by the API fall into several groups:

- * Registration. These functions allow an ASA to register itself with the GRASP core, and allow a registered ASA to register the GRASP objectives that it will manipulate.
- * Discovery. This function allows an ASA that needs to initiate negotiation or synchronization of a particular objective to discover a peer willing to respond.
- * Negotiation. These functions allow an ASA to act as an initiator (requester) or responder (listener) for a GRASP negotiation session. After initiation, negotiation is a symmetric process, so most of the functions can be used by either party.
- * Synchronization. These functions allow an ASA to act as an initiator (requester) or responder (listener and data source) for a GRASP synchronization session.
- * Flooding. These functions allow an ASA to send and receive an objective that is flooded to all nodes of the ACP.

Some example logic flows for a resource management ASA are given in [I-D.ietf-anima-asa-guidelines], which may be of help in understanding the following descriptions. The next section describes parameters and data structures used in multiple API calls. The following sections describe various groups of function APIs. Those APIs that do not list asynchronous mechanisms are implicitly synchronous in their behaviour.

2.3.2. Parameters and data structures

2.3.2.1. Integers

In this API, integers are assumed to be 32 bit unsigned integers (uint32_t) unless otherwise indicated.

2.3.2.2. Errorcode

All functions in the API have an unsigned 'errorcode' integer as their return value (the first return value in languages that allow multiple return values). An errorcode of zero indicates success. Any other value indicates failure of some kind. The first three errorcodes have special importance:

1. Declined: used to indicate that the other end has sent a GRASP Negotiation End message (M_END) with a Decline option (O_DECLINE).
2. No reply: used in non-blocking calls to indicate that the other end has sent no reply so far (see Section 2.2).
3. Unspecified error: used when no more specific error code applies.

Appendix A gives a full list of currently suggested error codes, based on implementation experience. While there is no absolute requirement for all implementations to use the same error codes, this is highly recommended for portability of applications.

2.3.2.3. Timeout

Wherever a 'timeout' parameter appears, it is an unsigned integer expressed in milliseconds. Except for the discover() function, if it is zero, the GRASP default timeout (GRASP_DEF_TIMEOUT, see [I-D.ietf-anima-grasp]) will apply. If no response is received before the timeout expires, the call will fail unless otherwise noted.

2.3.2.4. Objective

An 'objective' parameter is a data structure with the following components:

- * name (UTF-8 string) - the objective's name
- * neg (Boolean flag) - True if objective supports negotiation (default False)
- * synch (Boolean flag) - True if objective supports synchronization (default False)
- * dry (Boolean flag) - True if objective supports dry-run negotiation (default False)
 - Note 1: Only one of 'synch' or 'neg' may be True.
 - Note 2: 'dry' must not be True unless 'neg' is also True.
 - Note 3: In some programming languages the preferred implementation may be to represent the Boolean flags as bits in a single byte, which is how they are encoded in GRASP messages. In other languages an enumeration might be preferable.
- * loop_count (unsigned integer, uint8_t) - Limit on negotiation steps etc. (default GRASP_DEF_LOOPCT, see [I-D.ietf-anima-grasp]) The 'loop_count' is set to a suitable value by the initiator of a negotiation, to prevent indefinite loops. It is also used to limit the propagation of discovery and flood messages.
- * value - a specific data structure expressing the value of the objective. The format is language dependent, with the constraint that it can be validly represented in CBOR [RFC8949].

An important advantage of CBOR is that the value of an objective can be completely opaque to the GRASP core yet pass transparently through it to and from the ASA. Although the GRASP core must validate the format and syntax of GRASP messages, it cannot validate the value of an objective; all it can do is detect malformed CBOR. The handling of decoding errors depends on the CBOR library in use, but a corresponding error code ('CBORfail') is defined in the API and will be returned to the ASA if a faulty message can be assigned to a current GRASP session. However, it is the responsibility of each ASA to validate the value of a received objective, as discussed in Section 5.3 of [RFC8949]. If the programming language in use is suitably object-oriented, the GRASP API may deserialize the value and present it to the ASA as

an object. If not, it will be presented as a CBOR data item. In all cases, the syntax and semantics of the objective value are the responsibility of the ASA.

A requirement for all language mappings and all API implementations is that, regardless of what other options exist for a language-specific representation of the value, there is always an option to use a raw CBOR data item as the value. The API will then wrap this with CBOR Tag 24 as an encoded CBOR data item for transmission via GRASP, and unwrap it after reception. By this means, ASAs will be able to communicate regardless of programming language.

The 'name' and 'value' fields are of variable length. GRASP does not set a maximum length for these fields, but only for the total length of a GRASP message. Implementations might impose length limits.

An example data structure definition for an objective in the C language, using at least the C99 version, and assuming the use of a particular CBOR library [libcbor], is:

```
typedef struct {
    unsigned char *name;
    uint8_t flags;           // flag bits as defined by GRASP
    uint8_t loop_count;
    uint32_t value_size;     // size of value in bytes
    cbor_mutable_data cbor_value;
                          // CBOR bytestring (libcbor/cbor/data.h)
} objective;
```

An example data structure definition for an objective in the Python language (version 3.4 or later) is:

```
class objective:
    """A GRASP objective"""
    def __init__(self, name):
        self.name = name           #Unique name (string)
        self.negotiate = False    #True if objective supports negotiation
        self.dryrun = False       #True if objective supports dry-run neg.
        self.synch = False        #True if objective supports synch
        self.loop_count = GRASP_DEF_LOOPCT # Default starting value
        self.value = None         #Place holder; any valid Python object
```

2.3.2.5. ASA_locator

An 'ASA_locator' parameter is a data structure with the following contents:

- * locator - The actual locator, either an IP address or an ASCII string.
- * ifi (unsigned integer) - The interface identifier index via which this was discovered (of limited use to most ASAs).
- * expire (system dependent type) - The time on the local system clock when this locator will expire from the cache
- * The following cover all locator types currently supported by GRASP:
 - is_ipaddress (Boolean) - True if the locator is an IP address
 - is_fqdn (Boolean) - True if the locator is an FQDN
 - is_uri (Boolean) - True if the locator is a URI
 - These options are mutually exclusive. Depending on the programming language, they could be represented as a bit pattern or an enumeration.
- * diverted (Boolean) - True if the locator was discovered via a Divert option
- * protocol (unsigned integer) - Applicable transport protocol (IPPROTO_TCP or IPPROTO_UDP). These constants are defined in the CDDL specification of GRASP [I-D.ietf-anima-grasp].
- * port (unsigned integer) - Applicable port number

The 'locator' field is of variable length in the case of an FQDN or a URI. GRASP does not set a maximum length for this field, but only for the total length of a GRASP message. Implementations might impose length limits.

It should be noted that when one ASA discovers the ASA_locator of another, there is no explicit authentication mechanism. In accordance with the trust model provided by the secure ACP, ASAs are presumed to provide correct locators in response to discovery. See the section 'Locator Options' of [I-D.ietf-anima-grasp] for further details.

2.3.2.6. Tagged_objective

A 'tagged_objective' parameter is a data structure with the following contents:

- * objective - An objective
- * locator - The ASA_locator associated with the objective, or a null value.

2.3.2.7. Asa_handle

Although an authentication and authorization scheme for ASAs has not been defined, the API provides a very simple hook for such a scheme. When an ASA starts up, it registers itself with the GRASP core, which provides it with an opaque handle that, although not cryptographically protected, would be difficult for a third party to predict. The ASA must present this handle in future calls. This mechanism will prevent some elementary errors or trivial attacks such as an ASA manipulating an objective it has not registered to use.

Thus, in most calls, an 'asa_handle' parameter is required. It is generated when an ASA first registers with GRASP, and the ASA must then store the asa_handle and use it in every subsequent GRASP call. Any call in which an invalid handle is presented will fail. It is an up to 32-bit opaque value (for example represented as a uint32_t, depending on the language). Since it is only used locally, not in GRASP messages, it is only required to be unique within the local GRASP instance. It is valid until the ASA terminates. It should be unpredictable; a possible implementation is to use the same mechanism that GRASP uses to generate Session Identifiers (see Section 2.3.2.8).

2.3.2.8. Session_handle and Callbacks

In some calls, a 'session_handle' parameter is required. This is an opaque data structure as far as the ASA is concerned, used to identify calls to the API as belonging to a specific GRASP session (see Section 2.2.3). It will be provided as a parameter in callback functions. As well as distinguishing calls from different sessions, it also allows GRASP to detect and ignore calls from non-existent or timed-out sessions.

In an event loop implementation, callback functions (Section 2.2.1) may be supported for all API functions that involve waiting for a remote operation:

discover() whose callback would be discovery_received().

request_negotiate() whose callback would be negotiate_step_received().

`negotiate_step()` whose callback would be `negotiate_step_received()`.

`listen_negotiate()` whose callback would be `negotiate_step_received()`.

`synchronize()` whose callback would be `synchronization_received()`.

Further details of callbacks are implementation-dependent.

2.3.3. Registration

These functions are used to register an ASA, and the objectives that it modifies, with the GRASP module. In the absence of an authorization model, these functions are very simple but they will avoid multiple ASAs choosing the same name, and will prevent multiple ASAs manipulating the same objective. If an authorization model is added to GRASP, these API calls would need to be modified accordingly.

* `register_asa()`

All ASAs must use this call before issuing any other API calls.

- Input parameter:

name of the ASA (UTF-8 string)

- Return value:

errorcode (unsigned integer)

asa_handle (unsigned integer)

- This initialises state in the GRASP module for the calling entity (the ASA). In the case of success, an 'asa_handle' is returned which the ASA must present in all subsequent calls. In the case of failure, the ASA has not been authorized and cannot operate. The 'asa_handle' value is undefined.

* `deregister_asa()`

- Input parameters:

asa_handle (unsigned integer)

name of the ASA (UTF-8 string)

- Return value:

 errorcode (unsigned integer)

- This removes all state in the GRASP module for the calling entity (the ASA), and deregisters any objectives it has registered. Note that these actions must also happen automatically if an ASA exits.
- Note - the ASA name is strictly speaking redundant in this call, but is present to detect and reject erroneous deregistrations.

* register_objective()

ASAs must use this call for any objective whose value they need to transmit by negotiation, synchronization or flooding.

- Input parameters:

 asa_handle (unsigned integer)

 objective (structure)

 ttl (unsigned integer - default GRASP_DEF_TIMEOUT)

 discoverable (Boolean - default False)

 overlap (Boolean - default False)

 local (Boolean - default False)

- Return value:

 errorcode (unsigned integer)

- This registers an objective that this ASA may modify and transmit to other ASAs by flooding or negotiation. It is not necessary to register an objective that is only received by GRASP synchronization or flooding. The 'objective' becomes a candidate for discovery. However, discovery responses should not be enabled until the ASA calls listen_negotiate() or listen_synchronize(), showing that it is able to act as a responder. The ASA may negotiate the objective or send synchronization or flood data. Registration is not needed for "read-only" operations, i.e., the ASA only wants to receive synchronization or flooded data for the objective concerned.

- The 'ttl' parameter is the valid lifetime (time to live) in milliseconds of any discovery response generated for this objective. The default value should be the GRASP default timeout (GRASP_DEF_TIMEOUT, see [I-D.ietf-anima-grasp]).
 - If the parameter 'discoverable' is True, the objective is immediately discoverable. This is intended for objectives that are only defined for GRASP discovery, and which do not support negotiation or synchronization.
 - If the parameter 'overlap' is True, more than one ASA may register this objective in the same GRASP instance. This is of value for life cycle management of ASAs [I-D.ietf-anima-asa-guidelines] and must be used consistently for a given objective (always True or always False).
 - If the parameter 'local' is True, discovery must return a link-local address. This feature is for objectives that must be restricted to the local link.
 - This call may be repeated for multiple objectives.
- * deregister_objective()
- Input parameters:
 - asa_handle (unsigned integer)
 - objective (structure)
 - Return value:
 - errorcode (unsigned integer)
 - The 'objective' must have been registered by the calling ASA; if not, this call fails. Otherwise, it removes all state in the GRASP module for the given objective.

2.3.4. Discovery

* discover()

This function may be used by any ASA to discover peers handling a given objective.

- Input parameters:
 - asa_handle (unsigned integer)

objective (structure)

timeout (unsigned integer)

minimum_TTL (unsigned integer)

- Return values:

errorcode (unsigned integer)

locator_list (structure)

- This returns a list of discovered 'ASA_locator's for the given objective. An empty list means that no locators were discovered within the timeout. Note that this structure includes all the fields described in Section 2.3.2.5.
- The parameter 'minimum_TTL' must be greater than or equal to zero. Any locally cached locators for the objective whose remaining time to live in milliseconds is less than or equal to 'minimum_TTL' are deleted first. Thus 'minimum_TTL' = 0 will flush all entries. Note that this will not affect sessions already in progress using the deleted locators.
- If the parameter 'timeout' is zero, any remaining locally cached locators for the objective are returned immediately and no other action is taken. (Thus, a call with 'minimum_TTL' and 'timeout' both equal to zero is pointless.)
- If the parameter 'timeout' is greater than zero, GRASP discovery is performed, and all results obtained before the timeout in milliseconds expires are returned. If no results are obtained, an empty list is returned after the timeout. That is not an error condition. GRASP discovery is not a deterministic process. If there are multiple nodes handling an objective, none, some or all of them will be discovered before the timeout expires.
- Asynchronous Mechanisms:
 - o Threaded implementation: This should be called in a separate thread if asynchronous operation is required.

- o Event loop implementation: An additional in/out 'session_handle' parameter is used. If the 'errorcode' parameter has the value 2 ('noReply'), no response has been received so far. The 'session_handle' parameter must be presented in subsequent calls. A callback may be used in the case of a non-zero timeout.

2.3.5. Negotiation

Since the negotiation mechanism is different from a typical client/server exchange, Figure 2 illustrates the sequence of calls and GRASP messages in a negotiation. Note that after the first protocol exchange, the process is symmetrical, with negotiating steps strictly alternating between the two sides. Either side can end the negotiation. Also, the side that is due to respond next can insert a delay at any time, to extend the other side's timeout. This would be used, for example, if an ASA needed to negotiate with a third party before continuing with the current negotiation.

The loop count embedded in the objective that is the subject of negotiation is initialised by the ASA that starts a negotiation, and then decremented by the GRASP core at each step, prior to sending each M_NEGOTIATE message. If it reaches zero, the negotiation will fail and each side will receive an error code.

Initiator -----		Responder -----	
			listen_negotiate() \ Await request
request_negotiate()			
M_REQ_NEG	->	negotiate_step()	\ Open session,
	<-	M_NEGOTIATE	/ start negotiation
negotiate_step()			
M_NEGOTIATE	->	negotiate_step()	\ Continue
	<-	M_NEGOTIATE	/ negotiation
		...	
negotiate_wait()			\ Insert
M_WAIT	->		/ delay
negotiate_step()			
M_NEGOTIATE	->	negotiate_step()	\ Continue
	<-	M_NEGOTIATE	/ negotiation
negotiate_step()			
M_NEGOTIATE	->	end_negotiate()	\ End
	<-	M_END	/ negotiation
			\ Process results

Figure 2: Negotiation sequence

As the negotiation proceeds, each side will update the value of the objective in accordance with its particular semantics, defined in the specification of the objective. Although many objectives will have values that can be ordered, so that negotiation can be a simple bidding process, this is not a requirement.

Failure to agree, a timeout, or loop count exhaustion may all end a negotiation session, but none of these cases is a protocol failure.

* `request_negotiate()`

This function is used by any ASA to initiate negotiation of a GRASP objective as a requester (client).

- Input parameters:

`asa_handle` (unsigned integer)

`objective` (structure)

`peer` (ASA_locator)

`timeout` (unsigned integer)

- Return values:

`errorcode` (unsigned integer)

`session_handle` (structure) (undefined unless successful)

`proffered_objective` (structure) (undefined unless successful)

`reason` (string) (empty unless negotiation declined)

- This function opens a negotiation session between two ASAs. Note that GRASP currently does not support multi-party negotiation, which would need to be added as an extended function.
- The 'objective' parameter must include the requested value, and its loop count should be set to a suitable starting value by the ASA. If not, the GRASP default will apply.

- Note that a given negotiation session may or may not be a dry-run negotiation; the two modes must not be mixed in a single session.
- The 'peer' parameter is the target node; it must be an 'ASA_locator' as returned by discover(). If 'peer' is null, GRASP discovery is automatically performed first to find a suitable peer (i.e., any node that supports the objective in question).
- The 'timeout' parameter is described in Section 2.3.2.3.
- If the 'errorcode' return value is 0, the negotiation has successfully started. There are then two cases:
 1. The 'session_handle' parameter is null. In this case the negotiation has succeeded with one exchange of messages and the peer has accepted the request. The returned 'proffered_objective' contains the value accepted by the peer, which is therefore equal to the value in the requested 'objective'. For this reason, no session handle is needed, since the session has ended.
 2. The 'session_handle' parameter is not null. In this case negotiation must continue. The 'session_handle' must be presented in all subsequent negotiation steps. The returned 'proffered_objective' contains the first value proffered by the negotiation peer in the first exchange of messages; in other words it is a counter-offer. The contents of this instance of the objective must be used to prepare the next negotiation step (see negotiate_step() below) because it contains the updated loop count, sent by the negotiation peer. The GRASP code automatically decrements the loop count by 1 at each step, and returns an error if it becomes zero. Since this terminates the negotiation, the other end will experience a timeout, which will terminate the other end of the session.

This function must be followed by calls to 'negotiate_step' and/or 'negotiate_wait' and/or 'end_negotiate' until the negotiation ends. 'request_negotiate' may then be called again to start a new negotiation.

- If the 'errorcode' parameter has the value 1 ('declined'), the negotiation has been declined by the peer (M_END and O_DECLINE features of GRASP). The 'reason' string is then available for information and diagnostic use, but it may be a null string. For this and any other error code, an exponential backoff is recommended before any retry (see Section 4).
 - Asynchronous Mechanisms:
 - o Threaded implementation: This should be called in a separate thread if asynchronous operation is required.
 - o Event loop implementation: The 'session_handle' parameter is used to distinguish multiple simultaneous sessions. If the 'errorcode' parameter has the value 2 ('noReply'), no response has been received so far. The 'session_handle' parameter must be presented in subsequent calls.
 - Use of dry run mode: This must be consistent within a GRASP session. The state of the 'dry' flag in the initial request_negotiate() call must be the same in all subsequent negotiation steps of the same session. The semantics of the dry run mode are built into the ASA; GRASP merely carries the flag bit.
 - Special note for the ACP infrastructure ASA: It is likely that this ASA will need to discover and negotiate with its peers in each of its on-link neighbors. It will therefore need to know not only the link-local IP address but also the physical interface and transport port for connecting to each neighbor. One implementation approach to this is to include these details in the 'session_handle' data structure, which is opaque to normal ASAs.
- * listen_negotiate()

This function is used by an ASA to start acting as a negotiation responder (listener) for a given GRASP objective.

- Input parameters:
 - asa_handle (unsigned integer)
 - objective (structure)
- Return values:
 - errorcode (unsigned integer)

session_handle (structure) (undefined unless successful)

requested_objective (structure) (undefined unless successful)

- This function instructs GRASP to listen for negotiation requests for the given 'objective'. It also enables discovery responses for the objective, as mentioned under register_objective() in Section 2.3.3.
- Asynchronous Mechanisms:
 - o Threaded implementation: It will block waiting for an incoming request, so should be called in a separate thread if asynchronous operation is required. Unless there is an unexpected failure, this call only returns after an incoming negotiation request. If the ASA supports multiple simultaneous transactions, a new sub-thread must be spawned for each new session, so that listen_negotiate() can be called again immediately.
 - o Event loop implementation: A 'session_handle' parameter is used to distinguish individual sessions. If the ASA supports multiple simultaneous transactions, a new event must be inserted in the event loop for each new session, so that listen_negotiate() can be reactivated immediately.
- This call only returns (threaded model) or triggers (event loop) after an incoming negotiation request. When this occurs, 'requested_objective' contains the first value requested by the negotiation peer. The contents of this instance of the objective must be used in the subsequent negotiation call because it contains the loop count sent by the negotiation peer. The 'session_handle' must be presented in all subsequent negotiation steps.
- This function must be followed by calls to 'negotiate_step' and/or 'negotiate_wait' and/or 'end_negotiate' until the negotiation ends.
- If an ASA is capable of handling multiple negotiations simultaneously, it may call 'listen_negotiate' simultaneously from multiple threads, or insert multiple events. The API and GRASP implementation must support re-entrant use of the listening state and the negotiation calls. Simultaneous sessions will be distinguished by the threads or events themselves, the GRASP session handles, and the underlying unicast transport sockets.

* `stop_listen_negotiate()`

This function is used by an ASA to stop acting as a responder (listener) for a given GRASP objective.

- Input parameters:

`asa_handle` (unsigned integer)

`objective` (structure)

- Return value:

`errorcode` (unsigned integer)

- Instructs GRASP to stop listening for negotiation requests for the given objective, i.e., cancels 'listen_negotiate'.

- Asynchronous Mechanisms:

o Threaded implementation: Must be called from a different thread than 'listen_negotiate'.

o Event loop implementation: no special considerations.

* `negotiate_step()`

This function is used by either ASA in a negotiation session to make the next step in negotiation.

- Input parameters:

`asa_handle` (unsigned integer)

`session_handle` (structure)

`objective` (structure)

`timeout` (unsigned integer) as described in Section 2.3.2.3

- Return values:

Exactly as for 'request_negotiate'

- Executes the next negotiation step with the peer. The 'objective' parameter contains the next value being proffered by the ASA in this step. It must also contain the latest 'loop_count' value received from request_negotiate() or negotiate_step().
 - Asynchronous Mechanisms:
 - o Threaded implementation: Usually called in the same thread as the preceding 'request_negotiate' or 'listen_negotiate', with the same value of 'session_handle'.
 - o Event loop implementation: Must use the same value of 'session_handle' returned by the preceding 'request_negotiate' or 'listen_negotiate'.
- * negotiate_wait()
- This function is used by either ASA in a negotiation session to delay the next step in negotiation.
- Input parameters:
 - asa_handle (unsigned integer)
 - session_handle (structure)
 - timeout (unsigned integer)
 - Return value:
 - errorcode (unsigned integer)
 - Requests the remote peer to delay the negotiation session by 'timeout' milliseconds, thereby extending the original timeout. This function simply triggers a GRASP Confirm Waiting message (see [I-D.ietf-anima-grasp] for details).
 - Asynchronous Mechanisms:
 - o Threaded implementation: Called in the same thread as the preceding 'request_negotiate' or 'listen_negotiate', with the same value of 'session_handle'.
 - o Event loop implementation: Must use the same value of 'session_handle' returned by the preceding 'request_negotiate' or 'listen_negotiate'.

* `end_negotiate()`

This function is used by either ASA in a negotiation session to end a negotiation.

- Input parameters:

`asa_handle` (unsigned integer)

`session_handle` (structure)

`result` (Boolean)

`reason` (UTF-8 string)

- Return value:

`errorcode` (unsigned integer)

- End the negotiation session.

'result' = True for accept (successful negotiation), False for decline (failed negotiation).

'reason' = string describing reason for decline (may be null; ignored if accept).

- Asynchronous Mechanisms:

o Threaded implementation: Called in the same thread as the preceding 'request_negotiate' or 'listen_negotiate', with the same value of 'session_handle'.

o Event loop implementation: Must use the same value of 'session_handle' returned by the preceding 'request_negotiate' or 'listen_negotiate'.

2.3.6. Synchronization and Flooding

* `synchronize()`

This function is used by any ASA to cause synchronization of a GRASP objective as a requester (client).

- Input parameters:

`asa_handle` (unsigned integer)

objective (structure)

peer (ASA_locator)

timeout (unsigned integer)

- Return values:

errorcode (unsigned integer)

result (structure) (undefined unless successful)

- This call requests the synchronized value of the given 'objective'.
- If the 'peer' parameter is null, and the objective is already available in the local cache, the flooded objective is returned immediately in the 'result' parameter. In this case, the 'timeout' is ignored.
- If the 'peer' parameter is not null, or a cached value is not available, synchronization with a discovered ASA is performed. If successful, the retrieved objective is returned in the 'result' value.
- The 'peer' parameter is an 'ASA_locator' as returned by discover(). If 'peer' is null, GRASP discovery is automatically performed first to find a suitable peer (i.e., any node that supports the objective in question).
- The 'timeout' parameter is described in Section 2.3.2.3.
- This call should be repeated whenever the latest value is needed.
- Asynchronous Mechanisms:
 - o Threaded implementation: Call in a separate thread if asynchronous operation is required.
 - o Event loop implementation: An additional in/out 'session_handle' parameter is used, as in request_negotiate(). If the 'errorcode' parameter has the value 2 ('noReply'), no response has been received so far. The 'session_handle' parameter must be presented in subsequent calls.

- In the case of failure, an exponential backoff is recommended before retrying (Section 4).

* `listen_synchronize()`

This function is used by an ASA to start acting as a synchronization responder (listener) for a given GRASP objective.

- Input parameters:

`asa_handle` (unsigned integer)

`objective` (structure)

- Return value:

`errorcode` (unsigned integer)

- This instructs GRASP to listen for synchronization requests for the given objective, and to respond with the value given in the 'objective' parameter. It also enables discovery responses for the objective, as mentioned under `register_objective()` in Section 2.3.3.
- This call is non-blocking and may be repeated whenever the value changes.

* `stop_listen_synchronize()`

This function is used by an ASA to stop acting as a synchronization responder (listener) for a given GRASP objective.

- Input parameters:

`asa_handle` (unsigned integer)

`objective` (structure)

- Return value:

`errorcode` (unsigned integer)

- This call instructs GRASP to stop listening for synchronization requests for the given 'objective', i.e. it cancels a previous `listen_synchronize`.

* `flood()`

This function is used by an ASA to flood one or more GRASP objectives throughout the autonomic network.

Note that each GRASP node caches all flooded objectives that it receives, until each one's time-to-live expires. Cached objectives are tagged with their origin as well as an expiry time, so multiple copies of the same objective may be cached simultaneously. Further details are given in the section 'Flood Synchronization Message' of [I-D.ietf-anima-grasp]

- Input parameters:

asa_handle (unsigned integer)

ttl (unsigned integer)

tagged_objective_list (structure)

- Return value:

errorcode (unsigned integer)

- This call instructs GRASP to flood the given synchronization objective(s) and their value(s) and associated locator(s) to all GRASP nodes.
- The 'ttl' parameter is the valid lifetime (time to live) of the flooded data in milliseconds (0 = infinity)
- The 'tagged_objective_list' parameter is a list of one or more 'tagged_objective' couplets. The 'locator' parameter that tags each objective is normally null but may be a valid 'ASA_locator'. Infrastructure ASAs needing to flood an {address, protocol, port} 3-tuple with an objective create an ASA_locator object to do so. If the IP address in that locator is the unspecified address ('::') it is replaced by the link-local address of the sending node in each copy of the flood multicast, which will be forced to have a loop count of 1. This feature is for objectives that must be restricted to the local link.
- The function checks that the ASA registered each objective.
- This call may be repeated whenever any value changes.

* get_flood()

This function is used by any ASA to obtain the current value of a flooded GRASP objective.

- Input parameters:

asa_handle (unsigned integer)

objective (structure)

- Return values:

errorcode (unsigned integer)

tagged_objective_list (structure) (undefined unless successful)

- This call instructs GRASP to return the given synchronization objective if it has been flooded and its lifetime has not expired.

- The 'tagged_objective_list' parameter is a list of 'tagged_objective' couplets, each one being a copy of the flooded objective and a corresponding locator. Thus if the same objective has been flooded by multiple ASAs, the recipient can distinguish the copies.

- Note that this call is for advanced ASAs. In a simple case, an ASA can simply call synchronize() in order to get a valid flooded objective.

* expire_flood()

This function may be used by an ASA to expire specific entries in the local GRASP flood cache.

- Input parameters:

asa_handle (unsigned integer)

tagged_objective (structure)

- Return value:

errorcode (unsigned integer)

- This is a call that can only be used after a preceding call to get_flood() by an ASA that is capable of deciding that the flooded value is stale or invalid. Use with care.

- The 'tagged_objective' parameter is the one to be expired.

2.3.7. Invalid Message Function

* send_invalid()

This function may be used by any ASA to stop an ongoing GRASP session.

- Input parameters:

asa_handle (unsigned integer)

session_handle (structure)

info (bytes)

- Return value:

errorcode (unsigned integer)

- Sends a GRASP Invalid Message (M_INVALID) message, as described in [I-D.ietf-anima-grasp]. Should not be used if end_negotiate() would be sufficient. Note that this message may be used in response to any unicast GRASP message that the receiver cannot interpret correctly. In most cases this message will be generated internally by a GRASP implementation.

'info' = optional diagnostic data supplied by the ASA. May be raw bytes from the invalid message.

3. Implementation Status [RFC Editor: please remove]

A prototype open source Python implementation of GRASP, including an API similar to this document, has been used to verify the concepts for the threaded model. It may be found at <https://github.com/becarpenter/graspy> with associated documentation and demonstration ASAs.

4. Security Considerations

Security considerations for the GRASP protocol are discussed in [I-D.ietf-anima-grasp]. These include denial of service issues, even though these are considered a low risk in the ACP. In various places GRASP recommends an exponential backoff. An ASA using the API should use exponential backoff after failed discover(), req_negotiate() or synchronize() operations. The timescale for such backoffs depends on the semantics of the GRASP objective concerned. Additionally, a

flood() operation should not be repeated at shorter intervals than is useful. The appropriate interval depends on the semantics of the GRASP objective concerned. These precautions are intended to assist the detection of denial of service attacks.

As a general precaution, all ASAs able to handle multiple negotiation or synchronization requests in parallel may protect themselves against a denial of service attack by limiting the number of requests they handle simultaneously and silently discarding excess requests. It might also be useful for the GRASP core to limit the number of objectives registered by a given ASA, the total number of ASAs registered, and the total number of simultaneous sessions, to protect system resources. During times of high autonomic activity, such as recovery from widespread faults, ASAs may experience many GRASP session failures. Guidance on making ASAs suitably robust is given in [I-D.ietf-anima-asa-guidelines].

As noted earlier, the trust model is that all ASAs in a given autonomic network communicate via a secure autonomic control plane and therefore trust each other's messages. Specific authorization of ASAs to use particular GRASP objectives is a subject for future study, also briefly discussed in [I-D.ietf-anima-grasp].

The careful reader will observe that a malicious ASA could extend a negotiation session indefinitely by use of the negotiate_wait() function or by manipulating the loop count of an objective. A robustly implemented ASA could detect such behavior by a peer and break off negotiation.

The 'asa_handle' is used in the API as a first line of defence against a malware process attempting to imitate a legitimately registered ASA. The 'session_handle' is used in the API as a first line of defence against a malware process attempting to hijack a GRASP session. Both these handles are likely to be created using GRASP's 32-bit pseudo-random session ID. By construction, GRASP avoids the risk of session ID collisions (see the section 'Session Identifier' of [I-D.ietf-anima-grasp]). There remains a finite probability that an attacker could guess a session ID, session_handle, or asa_handle. However, this would only be of value to an attacker that had already penetrated the ACP, which would allow many other simpler forms of attack than hijacking GRASP sessions.

5. IANA Considerations

This document makes no request of the IANA.

6. Acknowledgements

Excellent suggestions were made by Ignas Bagdonas, Carsten Bormann, Laurent Ciavaglia, Roman Danyliw, Toerless Eckert, Benjamin Kaduk, Erik Kline, Murray Kucherawy, Paul Kyzivat, Guangpeng Li, Michael Richardson, Joseph Salowey, Eric Vyncke, Magnus Westerlund, Rob Wilton, and other participants in the ANIMA WG and the IESG.

7. References

7.1. Normative References

- [I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017, <<https://tools.ietf.org/html/draft-ietf-anima-grasp-15>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.

7.2. Informative References

- [I-D.ciavaglia-anima-coordination]
Ciavaglia, L. and P. Peloso, "Autonomic Functions Coordination", Work in Progress, Internet-Draft, draft-ciavaglia-anima-coordination-01, 21 March 2016, <<https://tools.ietf.org/html/draft-ciavaglia-anima-coordination-01>>.
- [I-D.ietf-anima-asa-guidelines]
Carpenter, B., Ciavaglia, L., Jiang, S., and P. Pierre, "Guidelines for Autonomic Service Agents", Work in Progress, Internet-Draft, draft-ietf-anima-asa-guidelines-00, 14 November 2020, <<https://tools.ietf.org/html/draft-ietf-anima-asa-guidelines-00>>.
- [I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft,

draft-ietf-anima-autonomic-control-plane-30, 30 October 2020, <<https://tools.ietf.org/html/draft-ietf-anima-autonomic-control-plane-30>>.

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

Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-bootstrapping-keyinfra-45, 11 November 2020, <<https://tools.ietf.org/html/draft-ietf-anima-bootstrapping-keyinfra-45>>.

[I-D.ietf-anima-grasp-distribution]

Liu, B., Xiao, X., Hecker, A., Jiang, S., Despotovic, Z., and B. Carpenter, "Information Distribution over GRASP", Work in Progress, Internet-Draft, draft-ietf-anima-grasp-distribution-01, 1 September 2020, <<https://tools.ietf.org/html/draft-ietf-anima-grasp-distribution-01>>.

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

Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", Work in Progress, Internet-Draft, draft-ietf-anima-reference-model-10, 22 November 2018, <<https://tools.ietf.org/html/draft-ietf-anima-reference-model-10>>.

[libcbor] Kalvoda, P., "libcbor - Documentation", December 2020, <<https://libcbor.readthedocs.io/>>.

Appendix A. Error Codes

This Appendix lists the error codes defined so far on the basis of implementation experience, with suggested symbolic names and corresponding descriptive strings in English. It is expected that complete API implementations will provide for localisation of these descriptive strings, and that additional error codes will be needed according to implementation details.

The error codes that may only be returned by one or two functions are annotated accordingly, and the others may be returned by numerous functions. The 'noSecurity' error will be returned to most calls if GRASP is running in an insecure mode (i.e., with no secure substrate such as the ACP), except for the specific DULL usage mode described in the section 'Discovery Unsolicited Link-Local' of [I-D.ietf-anima-grasp].

ok	0 "OK"
declined	1 "Declined" (req_negotiate, negotiate_step)
noReply	2 "No reply" (indicates waiting state in event loop calls)
unspec	3 "Unspecified error"
ASAFull	4 "ASA registry full" (register_asa)
dupASA	5 "Duplicate ASA name" (register_asa)
noASA	6 "ASA not registered"
notYourASA	7 "ASA registered but not by you" (deregister_asa)
notBoth	8 "Objective cannot support both negotiation and synchronization" (register_obj)
notDry	9 "Dry-run allowed only with negotiation" (register_obj)
notOverlap	10 "Overlap not supported by this implementation" (register_obj)
objFull	11 "Objective registry full" (register_obj)
objReg	12 "Objective already registered" (register_obj)
notYourObj	13 "Objective not registered by this ASA"
notObj	14 "Objective not found"
notNeg	15 "Objective not negotiable" (req_negotiate, listen_negotiate)
noSecurity	16 "No security"
noDiscReply	17 "No reply to discovery" (req_negotiate)
sockErrNegRq	18 "Socket error sending negotiation request" (req_negotiate)
noSession	19 "No session"
noSocket	20 "No socket"
loopExhausted	21 "Loop count exhausted" (negotiate_step)
sockErrNegStep	22 "Socket error sending negotiation step" (negotiate_step)
noPeer	23 "No negotiation peer" (req_negotiate, negotiate_step)
CBORfail	24 "CBOR decode failure" (req_negotiate, negotiate_step, synchronize)
invalidNeg	25 "Invalid Negotiate message" (req_negotiate, negotiate_step)
invalidEnd	26 "Invalid end message" (req_negotiate, negotiate_step)
noNegReply	27 "No reply to negotiation step" (req_negotiate, negotiate_step)
noValidStep	28 "No valid reply to negotiation step" (req_negotiate, negotiate_step)
sockErrWait	29 "Socket error sending wait message" (negotiate_wait)

sockErrEnd	30	"Socket error sending end message" (end_negotiate, send_invalid)
IDclash	31	"Incoming request Session ID clash" (listen_negotiate)
notSynch	32	"Not a synchronization objective" (synchronize, get_flood)
notFloodDisc	33	"Not flooded and no reply to discovery" (synchronize)
sockErrSynRq	34	"Socket error sending synch request" (synchronize)
noListener	35	"No synch listener" (synchronize)
noSynchReply	36	"No reply to synchronization request" (synchronize)
noValidSynch	37	"No valid reply to synchronization request" (synchronize)
invalidLoc	38	"Invalid locator" (flood)

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

draft-ietf-anima-grasp-api-10, 2021-01:

- * Closed two final IESG comments

draft-ietf-anima-grasp-api-09, 2020-12:

- * Added short discussions of CBOR usage and verification.
- * Added section on session termination.
- * Clarified that integers are uint32 or uint8.
- * Minor technical correction to timeout specification.
- * Clarified sequencing of negotiation messages.
- * Minor technical addition to request_negotiate() and synchronize() in event loop model.
- * Expanded several points in Security Considerations, including precautions against resource exhaustion.
- * Other clarifications and minor reorganizations; removed some duplicated text.
- * Updated references.

draft-ietf-anima-grasp-api-08, 2020-11:

- * Clarified trust model
- * Added explanations of GRASP objectives and sessions
- * Added note about non-idempotent messages
- * Added overview of API functions, and annotated each function with a brief description
- * Added protocol diagram for negotiation session
- * Clarified (absence of) authorization model
- * Changed precise semantics of synchronize() for flooded objectives
- * Clarified caching of flooded objectives
- * Changed 'age_limit' to 'minimum_TTL'
- * Improved security considerations, including DOS precautions
- * Annotated error codes to indicate which functions generate which errors
- * Other clarifications from Last Call reviews

draft-ietf-anima-grasp-api-07, 2020-10-13:

- * Improved diagram and its description
- * Added pointer to example logic flows
- * Added note on variable length parameters
- * Clarified that API decrements loop count automatically
- * Other corrections and clarifications from AD review

draft-ietf-anima-grasp-api-06, 2020-06-07:

- * Improved diagram
- * Numerous clarifications and layout changes

draft-ietf-anima-grasp-api-05, 2020-05-08:

- * Converted to xml2rfc v3

- * Editorial fixes.

draft-ietf-anima-grasp-api-04, 2019-10-07:

- * Improved discussion of layering, mentioned daemon.
- * Added callbacks and improved description of asynchronous operations.
- * Described use case for 'session_handle'.
- * More explanation of 'asa_handle'.
- * Change 'discover' to use 'age_limit' instead of 'flush'.
- * Clarified use of 'dry run'.
- * Editorial improvements.

draft-ietf-anima-grasp-api-03, 2019-01-21:

- * Replaced empty "logic flows" section by "implementation status".
- * Minor clarifications.
- * Editorial improvements.

draft-ietf-anima-grasp-api-02, 2018-06-30:

- * Additional suggestion for event-loop API.
- * Discussion of error code values.

draft-ietf-anima-grasp-api-01, 2018-03-03:

- * Editorial updates

draft-ietf-anima-grasp-api-00, 2017-12-23:

- * WG adoption
- * Editorial improvements.

draft-liu-anima-grasp-api-06, 2017-11-24:

- * Improved description of event-loop model.
- * Changed intended status to Informational.

- * Editorial improvements.

draft-liu-anima-grasp-api-05, 2017-10-02:

- * Added send_invalid()

draft-liu-anima-grasp-api-04, 2017-06-30:

- * Noted that simple nodes might not include the API.

- * Minor clarifications.

draft-liu-anima-grasp-api-03, 2017-02-13:

- * Changed error return to integers.

- * Required all implementations to accept objective values in CBOR.

- * Added non-blocking alternatives.

draft-liu-anima-grasp-api-02, 2016-12-17:

- * Updated for draft-ietf-anima-grasp-09

draft-liu-anima-grasp-api-02, 2016-09-30:

- * Added items for draft-ietf-anima-grasp-07

- * Editorial corrections

draft-liu-anima-grasp-api-01, 2016-06-24:

- * Updated for draft-ietf-anima-grasp-05

- * Editorial corrections

draft-liu-anima-grasp-api-00, 2016-04-04:

- * Initial version

Authors' Addresses

Brian Carpenter
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

Wendong Wang
BUPT University
Beijing University of Posts & Telecom.
No.10 Xitucheng Road
Hai-Dian District, Beijing 100876
P.R. China

Email: wdwang@bupt.edu.cn

Xiangyang Gong
BUPT University
Beijing University of Posts & Telecom.
No.10 Xitucheng Road
Hai-Dian District, Beijing 100876
P.R. China

Email: xygong@bupt.edu.cn

ANIMA WG
INTERNET-DRAFT
Intended Status: Standard Track
Expires: June 14, 2020

B. Liu
Huawei Technologies
X. Xiao
A. Hecker
MRC, Huawei Technologies
S. Jiang
Huawei Technologies
Z. Despotovic
MRC, Huawei Technologies
December 12, 2019

Information Distribution in Autonomic Networking
draft-liu-anima-grasp-distribution-13

Abstract

This document proposes a solution for information distribution in autonomic networks. Information distribution is categorized into two different modes: 1) instantaneous distribution; 2) publication for retrieval. In the former case, the information is sent, propagates and is disposed of after reception. In the latter case, information needs to be stored in the network.

The capabilities to distribute information are basic and fundamental needs for an autonomous network (cf. ANI [I-D.ietf-anima-reference-model]). This document describes typical use cases of information distribution in ANI and requirements to ANI, such that rich information distribution can be natively supported. The document proposes extensions to the autonomic nodes and suggests an implementation based on GRASP extensions as a protocol on the wire.

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/lid-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	3
3.	Requirements of Enriched Information Distribution	4
4.	Node Behaviors	5
4.1	Instant Information Distribution (IID) Sub-module	5
4.2	Asynchronous Information Distribution (AID) Sub-module	6
4.3	Summary	10
5.	Extending GRASP for Information Distribution	10
5.1	Realizing Instant P2P Transmission	10
5.2	Realizing Instant Selective Flooding	11
5.3	Realizing Subscription as An Event	11
5.4	Un_Subscription Objective Option	12
5.5	Publishing Objective Option	12
6.	Security Considerations	13
7.	IANA Considerations	13
8.	References	13
8.1	Normative References	13
8.2	Informative References	13
	Authors' Addresses	14
	Appendix A. Real-world Use Cases of Information Distribution	15
	Appendix B. Information Distribution Module in ANI	18
	Appendix C. Asynchronous Information Distribution Integrated with GRASP APIs	18

1 Introduction

In an autonomic network, autonomic functions (AFs) running on autonomic nodes constantly exchange information, e.g. AF control/management signaling or AF data exchange. This document discusses the information distribution capability of such exchanges between AFs.

Depending on the number of participants, the information can be distributed in the following scenarios:

- 1) Point-to-point (P2P) Communication: information is exchanged between parties, i.e. two nodes.
- 2) One-to-Many Communication: information exchanges involve an information source and multiple receivers.

The approaches to information distribution can be chiefly categorized into two basic modes:

- 1) An instantaneous mode (push): a source sends the actual content (e.g. control/management signaling, synchronization data and so on) to all interested receiver(s) immediately. Generally, some preconfiguration is required, as nodes interested in this information must be already known to all nodes in the sense that any receiving node must be able to decide, to which nodes this data is to be sent.
- 2) An asynchronous mode (delayed pull): here, a source publishes the content in some form in the network, which may later be looked for, found and retrieved by some endpoints in the AN. Here, depending on the size of the content, either the whole content or only its metadata might be published into the AN. In the latter case the metadata (e.g. a content descriptor, e.g. a key, and a location in the ANI) may be used for the actual retrieval. Importantly, the source, i.e. here publisher, needs to be able to determine the node, where the information (or its metadata) can be stored.

To avoid repetitive implementations by each AF developer, this document opts for a common support for information distribution implemented as a basic ANI capability, therefore available to all AFs. In fact, GRASP already provides part of the capabilities.

Regardless, an AF may still define and implement its own information distribution capability. Such a capability may then be advertised using the common information distribution capability defined in this document. Overall, ANI nodes and AFs may decide, which of the

information distribution mechanisms they want to use for which type of information, according to their own preferences (e.g. semantic routing table, etc.)

This document first analyzes requirements for information distribution in autonomic networks (Section 3) and then discuss the relevant node behavior (Section 4). After that, the required GRASP extensions are formally introduced (Section 5).

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 for Information Distribution in ANI

The question of information distribution in an autonomic network can be discussed through particular use cases or more generally. Depending on the situation it can be quite simple or might require more complex provisions.

Indeed, in the simplest case, the information can be sent:

- 1) at once (in one packet, in one flow),
- 2) straightaway (send-and-forget),
- 3) to all nodes.

Presuming 1), 2) and 3) hold, information distribution in smaller or scarce topologies can be implemented using broadcast, i.e. unconstrained flooding. For reasons well-understood, this approach has its limits in larger and denser networks. In this case, a graph can be constructed such that it contains every node exactly once (e.g. a spanning tree), still allowing to distribute any information to all nodes straightaway. Multicast tree construction protocols could be used in this case. There are reasonable use cases for such scenarios, as presented in Appendix B.

A more complex scenario arises, if only 1) and 2) hold, but the information only concerns a subset of nodes. Then, some kind of selection becomes required, to which nodes the given information should be distributed. Here, a further distinction is necessary; notably, if the selection of the target nodes is with respect to the nature or position of the node, or whether it is with respect to the information content. If the first, some knowledge about the node types, its topological position, etc (e.g. the routing information within ANI) can be used to distinguish nodes accordingly. For instance, edge nodes and forwarding nodes can be distinguished in this way. If the distribution scope is primarily to be defined by the information elements, then a registration / join / subscription or label distribution mechanism is unavoidable. This would be the case, for instance, if the AFs can be dynamically deployed on nodes, and the information is majorily destined to the AFs. Then, depending on the current AF deployment, the distribution scope must be adjusted as well.

If only 1) holds, but the information content might be required again and again, or might not yet be fully available, then more complex mechanisms might be required to store the information within the network for later, for further redistribution, and for notification of interested nodes. Examples for this include distribution of reconfiguration information for different AF instances, which might not require an immediate action, but only an eventual update of the parameters. Also, in some situations, there could be a significant

delay between the occurrence of a new event and the full content availability (e.g. if the processing requires a lot of time).

Finally, none of the three might hold. Then, along with the subscription and notification, the actual content might be different from its metadata, i.e. some description of the content and, possibly, its location. The fetching can then be implemented in different, appropriate ways, if necessary as a complex transport session.

In essence, as flooding is usually not an option, and the interest of nodes for particular information elements can change over time, ANI should support autonomies also for the information distribution.

This calls for autonomic mechanisms in the ANI, allowing participating nodes to 1) advertise or publish 2) look for or subscribe to 3) store 4) fetch/retrieve 5) instantaneously push information elements.

In the following cases, situations depicting diverse information distribution needs are discussed.

1) Long Communication Intervals. The actual sending of the information is not necessarily instantaneous with some event. Advanced AFs may involve into longer jobs/tasks (e.g. database lookup, authentication etc.) when processing requests, and might not be able to reply immediately. Instead of actively waiting for the reply, a better way for an interested AF might be to get notified, when the reply is finally available.

2) Common Interest Distribution. AFs may share interest in common information. For example, the network intent will be distributed to network nodes enrolled, which is usually one-to-many scenario. Intent distribution can also be performed by an instant flooding (e.g. via GRASP) to every network node. However, because of network dynamics, not every node can be just ready at the moment when the network intent is broadcast. Also, a flooding often does not cover all network nodes as there is usually a limitation on the hop number. In fact, 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 consume but also generate data information. An example is 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.

Therefore, for ANI, in order to support various communication scenarios, an information distribution module is required, and both instantaneous and asynchronous communication models should be supported. Some real-world use cases are introduced in Appendix A.

4. Node Behaviors

In this section, how a node should behave in order to support the two identified modes of information distribution is discussed. An ANI is a distributed system, so the information distribution module must be implemented in a distributed way as well.

4.1 Instant Information Distribution (IID) Sub-module

In this case, An information sender directly specifies the information receiver(s). The instant information distribution sub-module will be the main element.

4.1.1 Instant P2P Communication

IID sub-module performs instant information transmission for ASAs running in an ANI. In specific, IID sub-module will have to retrieve the address of the information receiver specified by an ASA, then deliver the information to the receiver. Such a delivery can be done either in a connectionless or a connection-oriented way.

Current GRASP provides the capability to support instant P2P synchronization for ASAs. A P2P synchronization is a use case of P2P information transmission. However, as mention in Section 3, there are some scenarios where one node needs to transmit some information to another node(s). This is different to synchronization because after transmitting the information, the local status of the information does not have to be the same as the information sent to the receiver. This is not directly support by existing GRASP.

4.1.2 Instant Flooding Communication

IID sub-module finishes instant flooding for ASAs in an ANI. Instant flooding is for all ASAs in an ANI. An information sender has to specify a special destination address of the information and broadcast to all interfaces to its neighbors. When another IID sub-module receives such a broadcast, after checking its TTL, it further broadcast the message to the neighbors. In order to avoid flooding storms in an ANI, usually a TTL number is specified, so that after a pre-defined limit, the flooding message will not be further broadcast again.

In order to avoid unnecessary flooding, a selective flooding can be done where an information sender wants to send information to multiple receivers at once. When doing this, sending information needs to contain criteria to judge on which interfaces the distributed information should and should not be sent. Specifically, the criteria contain:

- o Matching Condition: a set of matching rules such as addresses of recipients, node features and so on.
- o Action: what the node needs to do when the Matching Condition is fulfilled. For example, the action could be forwarding or discarding the distributed message.

Sent information must be included in the message distributed from the sender. The receiving node reacts by first checking the carried Matching Condition in the message to decide who should consume the message, which could be either the node itself, some neighbors or both. If the node itself is a recipient, Action field is followed; if a neighbor is a recipient, the message is sent accordingly.

An exemplary extension to support selective flooding on GRASP is described in Section 5.

4.2 Asynchronous Information Distribution (AID) Sub-module

In asynchronous information distribution, sender(s) and receiver(s) are not immediately specified while they may appear in an asynchronous way. Firstly, AID sub-module enables that the information can be stored in the network; secondly, AID sub-module provides an information publication and subscription (Pub/Sub) mechanism for ASAs.

As 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. Details of the

two modules are described in the following sections.

4.2.1 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 hash value) 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. The IS module performs various checks of the request (e.g. permitted information size).
- 2) Storing Peer Mapping. The information block will be handled by the IS module in order to calculate/map to a peer node in the network. Since ANIMA network is a peer-to-peer network, a typical way is to use distributed 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) Storing Peer 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 requested storing peer will send feedback the source IS module.
- 4) Storing Peer Negotiation Response. Negotiation response from the storing peer 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 storing peer must be discovered (i.e. going to step 7).
- 5) Information Block Transfer. Before sending the information block to the storing peer that already 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, a bulk data transmission is needed. For that, there are multiple ways to

do it.

The first option is to utilize one of existing protocols that is independent of the GRASP stack. For example, a session connectivity can be established to the storing peer, and over the connection the bulky data can be transmitted part by part. In this case, the IS module should support basic TCP-based session protocols such as HTTP(s) or native TCP.

The second option is to directly use GRASP itself for bulky data transferring. [I-D.carpenter-anima-grasp-bulk-04].

6) Information Writing. Once the information block (or a smaller block) is received, the IS module of the storing peer will store the data block in the local storage is accessible.

7) (Optional) New Storing Peer Discovery. If the previously selected storing peer 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 ANI 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 known 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) Storing Peer Mapping. IS module maps the key/index of the requested information to a peer that stores the information, and prepares the information request. 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 storing peer and asks if such an information object is available.

4) Retrieval Negotiation Response. The storing peer checks the key/index of the information in the request, 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 originally identified storing peer, the source IS module will have to discover the location of the requested information.

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.

4.2.2 Event Queue The Event Queue (EQ) module is to help ASAs to publish information to the network and subscribe to interested information in asynchronous scenarios. In an ANI, information generated on network nodes is an event labeled 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: An EQ module provides isolated queues for different event groups. If two groups of AFs could have completely different purposes, the 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 can have different priorities in ANI. This corresponds to how much important or urgent the event implies. Some of them are more urgent than regular ones. Prioritization allows AFs to differentiate events (i.e. information) they publish or subscribe to.

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 EQ module on every network node operates 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 to be processed. By considering both aspects, the priority of the event will be determined.

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 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.

The category of event priority is defined as the following. In general, there are two event types:

1) Network Control Event: This type of events are defined by the ANI for operational purposes on network control. A pre-defined priority levels for required system messages is suggested. For highest level to lowest level, the priority value ranges from NC_PRIOR_HIGH to NC_PRIOR_LOW as integer values. The NC_PRIOR_* values will be defined later according to the total number system events required by the ANI;

2) Custom ASA Event: This type of events are defined by the ASAs of users. This specifies the priority of the message within a group of ASAs, therefore it is only effective among ASAs that join the same message group. Within the message group, a group header/leader has to define a list of priority levels ranging from CUST_PRIOR_HIGH to CUST_PRIOR_LOW. Such a definition completely depends on the individual purposes of the message group.

When a system message is delivered, its event type and event priority value have to be both specified;

Event contains the address where the information is stored, after a subscriber is notified, it directly retrieves the information from the given location.

4.3 Summary

In summary, the general requirements for the information distribution module on each autonomic node are realized by two sub-modules handling instant communications and asynchronous communications, respectively. For instantaneous mode, node requirements are simple,

calling for support for additional signaling. With minimum efforts, reusing the existing GRASP is possible.

For asynchronous mode, information distribution module uses new primitives on the wire, and implements an event queue and an information storage mechanism. An architectural consideration on ANI with the information distribution module is briefly discussed in Appendix B.

5. Extending GRASP for Information Distribution

5.1 Realizing Instant P2P Transmission

This could be a new message in GRASP. In fragmentary CDDL, an Un-solicited Synchronization message follows the pattern:

```
unsolicited_synch-message = [M_UNSOLIDSYNCH, 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).

5.2 Realizing Instant Selective Flooding

Since normal flooding is already supported by GRASP, this section only defines the selective flooding extension.

In fragmentary CDDL, the selective flooding follows the pattern:

```
selective-flood-option = [O_SELECTIVE_FLOOD, +O_MATCH-CONDITION,
                           match-object, action]
```

```
O_MATCH-CONDITION = [O_MATCH-CONDITION, Obj1, match-rule, Obj2]
Obj1 = text
```

```
match-rule = GREATER / LESS / WITHIN / CONTAIN
```

```
Obj2 = text
```

```
match-object = NEIGHBOR / SELF
```

```
action = FORWARD / DROP
```

The option field 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.

5.3 Realizing Subscription as An Event

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.

5.4 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.

5.5 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.

6. Security Considerations

The distribution source authentication could be done at multiple layers:

- o 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.
- o 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.

7. IANA Considerations

TBD.

8. References

8.1 Normative References

[I-D.ietf-anima-grasp]
Bormann, D., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-animagrasp-15 (Standard Track), October 2017.

8.2 Informative References

[RFC7575] Behringer, M., "Autonomic Networking: Definitions and Design Goals", RFC 7575, June 2015

[I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-behringer-anima-autonomic-control-plane-13, December 2017.

[I-D.ietf-anima-stable-connectivity-10]

Eckert, T., Behringer, M., "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-ietf-anima-stable-connectivity-10, February 2018.

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

Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Pierre P., Liu, B., Nobre, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-05, October 2017.

[I-D.du-anima-an-intent]

Du, Z., Jiang, S., Nobre, J., Ciavaglia, L., and M. Behringer, "ANIMA Intent Policy and Format", draft-duanima-an-intent-05 (work in progress), February 2017.

[I-D.ietf-anima-grasp-api]

Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", draft-ietf-anima-grasp-api-00 (work in progress), December 2017.

[I-D.carpenter-anima-grasp-bulk-04]

Carpenter, B., Jiang, S., Liu, B., "Transferring Bulk Data over the GeneRIC Autonomic Signaling Protocol (GRASP)", draft-carpenter-anima-grasp-bulk-04 (work in progress), July 3, 2019

[3GPP.29.500]

3GPP, "Technical Realization of Service Based Architecture", 3GPP TS 29.500 15.1.0, 09 2018

[3GPP.23.501]

3GPP, "System Architecture for the 5G System", 3GPP TS 23.501 15.2.0, 6 2018, <<http://www.3gpp.org/ftp/Specs/html-info/23501.htm>>.

[3GPP.23.502]

3GPP, "Procedures for the 5G System", 3GPP TS 23.502 15.2.0, 6 2018, <<http://www.3gpp.org/ftp/Specs/html-info/23502.htm>>.

[5GAA.use.cases]

White Paper "Toward fully connected vehicles: Edge computing for advanced automotive communications", 5GAA <<http://5gaa.org/news/toward-fully-connected-vehicles-edge-computing-for-advanced-automotive-communications/>>

Authors' Addresses

Bing Liu
Huawei Technologies
Q27, Huawei Campus

No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com

Sheng Jiang
Huawei Technologies
Q27, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: jiangsheng@huawei.com

Xun Xiao
Munich Research Center
Huawei technologies
Riesstr. 25, 80992, Muenchen, Germany

Emails: xun.xiao@huawei.com

Artur Hecker
Munich Research Center
Huawei technologies
Riesstr. 25, 80992, Muenchen, Germany

Emails: artur.hecker@huawei.com

Zoran Despotovic
Munich Research Center
Huawei technologies
Riesstr. 25, 80992, Muenchen, Germany

Emails: zoran.despotovic@huawei.com

Appendix A. Real-world Use Cases of Information Distribution

The requirement analysis in Section 3 shows that generally information distribution should be better off 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 5G mobile networks from 3GPP has 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. A concrete example is that a session management function (SMF) can subscribe to the REGISTER notification from an access management function (AMF) if there is a new user equipment 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. Specifically, 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 adopted 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

Connected car is one of 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, 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.

Appendix B. Information Distribution Module in ANI

This section describes how the information distribution module fits into the ANI and what extensions of GRASP are required [I-D.ietf-anima-grasp].

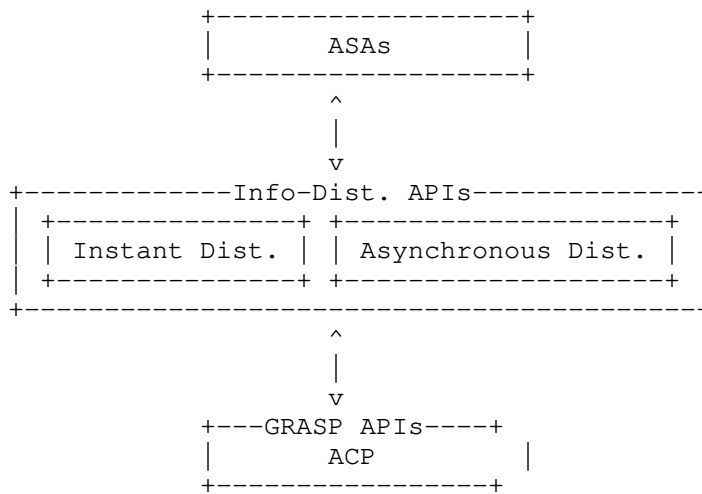


Figure 1. Information Distribution Module and GRASP Extension.

As the Fig 1 shows, the information distribution module two sub-modules for instant and asynchronous information distributions, respectively, and provides APIs to ASAs. Specific Behaviors of modules are described in Section 5.

Appendix C. Asynchronous ID Integrated with GRASP APIs

Actions triggered to the information distribution module will eventually invoke underlying GRASP APIs. Moreover, EQ and IS modules are usually correlated. When an AF(ASA) publishes information, not only such an event is translated and sent to EQ module, but also the information is indexed and stored simultaneously. Similarly, when an AF(ASA) 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.

o Storing and publishing information: This action involves both IS and EQ modules where a node that can store the information will be discovered first and related event will be published to the network. For this, GRASP APIs `discover()`, `synchronize()` and `flood()` are combined to compose such a procedure. In specific, `discover()` call will specify its objective being to "store_data" and the return parameters could be either an `ASA_locator` who will accept to store the data, or an error code indicating that no one could afford such data; after that, `synchronize()` call will send the data to the specified `ASA_locator` and the data will be stored at that node, with return of processing results like `store_data_ack`; meanwhile, such a successful event (i.e. data is stored successfully) will be flooded via a `flood()` call to interesting parties (such a multicast group existed).

o Subscribing and getting information: This action involves both IS and EQ modules as well where a node that is interested in a topic will subscribe the topic by triggering EQ module and if the topic is ready IS module will retrieve the content of the topic (i.e. the data). GRASP APIs such as `register_objective()`, `flood()`, `synchronize()` are combined to compose the procedure. In specific, any subscription action received by EQ module will be translated to `register_objective()` call where the interested topic will be the parameter inside of the call; the registration will be (selectively) flooded to the network by an API call of `flood()` with the option we extended in this draft; once a matched topic is found (because of the previous procedure), the node finding such a match will call API `synchronize()` to send the stored data to the subscriber.