ANIMA
Internet-Draft
Intended status: Standards Track
Expires: October 8, 2016

M. Behringer
Cisco Systems

April 6, 2016

An Autonomic IPv6 Addressing Scheme
draft-behringer-anima-autonomic-addressing-03

Abstract

The content of this document was merged into [I-D.ietf-anima-autonomic-control-plane]. This document it no longer maintained.

Table of Contents

1. Introduction

The content of this document was merged into [I-D.ietf-anima-autonomic-control-plane]. This document it no longer maintained.

2. Security Considerations

n/a

3. References

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-02 (work in progress), March 2016.

Author's Address

Michael H. Behringer
Cisco Systems
Building D, 45 Allee des Ormes
Mougins 06250
France

Email: mbehring@cisco.com

ANIMA
Internet-Draft
Intended status: Informational
Expires: April 18, 2016

M. Behringer, Ed.
Cisco Systems
B. Carpenter
Univ. of Auckland
T. Eckert
Cisco
L. Ciavaglia
Alcatel Lucent
B. Liu
Huawei Technologies
J. Nobre
Federal University of Rio Grande do Sul
J. Strassner
Huawei Technologies

October 16, 2015

A Reference Model for Autonomic Networking
draft-behringer-anima-reference-model-04

Abstract

This document describes a reference model for Autonomic Networking. The goal is to define how the various elements in an autonomic context work together, to describe their interfaces and relations. While the document is written as generally as possible, the initial solutions are limited to the chartered scope of the WG.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The document "Autonomic Networking - Definitions and Design Goals" [RFC7575] explains the fundamental concepts behind Autonomic Networking, and defines the relevant terms in this space. In section 5 it describes a high level reference model. This document defines this reference model with more detail, to allow for functional and protocol specifications to be developed in an architecturally consistent, non-overlapping manner. While the document is written as generally as possible, the initial solutions are limited to the chartered scope of the WG.

As discussed in [RFC7575], the goal of this work is not to focus exclusively on fully autonomic nodes or networks. In reality, most networks will run with some autonomic functions, while the rest of the network is traditionally managed. This reference model allows for this hybrid approach.

This is a living document and will evolve with the technical solutions developed in the ANIMA WG. Sections marked with (*) do not represent current charter items. While this document must give a long term architectural view, not all functions will be standardized at the same time.

2. The Network View

This section describes the various elements in a network with autonomic functions, and how these entities work together, on a high level. Subsequent sections explain the detailed inside view for each of the autonomic network elements, as well as the network functions (or interfaces) between those elements.

Figure 1 shows the high level view of an Autonomic Network. It consists of a number of autonomic nodes, which interact directly with each other. Those autonomic nodes provide a common set of capabilities across the network, called the "Autonomic Networking Infrastructure" (ANI). The ANI provides functions like naming, addressing, negotiation, synchronization, discovery and messaging.

Autonomic functions typically span several, possibly all nodes in the network. The atomic entities of an autonomic function are called the "Autonomic Service Agents" (ASA), which are instantiated on nodes.

```
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:            :       Autonomic Function 1        :                 :
: ASA 1      :      ASA 1      :      ASA 1      :          ASA 1  :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
             :                 :                 :
             :   +- - - - - - - - - - - - - - +  :
             :   :   Autonomic Function 2     :  :
             :   :  ASA 2      :      ASA 2   :  :
             :   +- - - - - - - - - - - - - - +  :
             :                 :                 :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:                Autonomic Networking Infrastructure               :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
+--------+   :    +--------+   :    +--------+   :        +--------+
| Node 1 |--------| Node 2 |--------| Node 3 |----...-----| Node n |
+--------+   :    +--------+   :    +--------+   :        +--------+
```

Figure 1: High level view of an Autonomic Network

In a horizontal view, autonomic functions span across the network, as well as the Autonomic Networking Infrastructure. In a vertical view, a node always implements the ANI, plus it may have one or several Autonomic Service Agents.

The Autonomic Networking Infrastructure (ANI) therefore is the foundation for autonomic functions. The current charter of the ANIMA WG is to specify the ANI, using a few autonomic functions as use cases.

3. The Autonomic Network Element

3.1. Architecture

This section describes an autonomic network element and its internal architecture. The reference model explained in the document "Autonomic Networking - Definitions and Design Goals" [RFC7575] shows the sources of information that an autonomic service agent can leverage: Self-knowledge, network knowledge (through discovery), Intent, and feedback loops. Fundamentally, there are two levels inside an autonomic node: the level of Autonomic Service Agents, and the level of the Autonomic Networking Infrastructure, with the former using the services of the latter. Figure 2 illustrates this concept.

```
+------------------------------------------------------------+
|                                                            |
| +-----------+        +------------+        +------------+  |
| | Autonomic |        | Autonomic  |        | Autonomic  |  |
| | Service   |        | Service    |        | Service    |  |
| | Agent 1   |        | Agent 2    |        | Agent 3    |  |
| +-----------+        +------------+        +------------+  |
|       ^                    ^                     ^         |
| -  -  | -  - API level -  -| -  -  -  -  -  -  - |-  -  -  |
|       V                    V                     V         |
|------------------------------------------------------------|
| Autonomic Networking Infrastructure                        |
|    - Data structures (ex: certificates, peer information)  |
|    - Autonomic Control Plane                               |
|    - Autonomic Node Addressing                             |
|    - Discovery, negotiation and synchronisation functions  |
|    - Intent distribution                                   |
|    - Aggregated reporting and feedback loops               |
|    - Routing                                               |
|------------------------------------------------------------|
|             Basic Operating System Functions               |
+------------------------------------------------------------+
```

Figure 2: Model of an autonomic node

The Autonomic Networking Infrastructure (lower part of Figure 2) contains node specific data structures, for example trust information about itself and its peers, as well as a generic set of functions, independent of a particular usage. This infrastructure should be generic, and support a variety of Autonomic Service Agents (upper part of Figure 2). The Autonomic Control Plane is the summary of all interactions of the Autonomic Networking Infrastructure with other nodes and services.

The use cases of "Autonomics" such as self-management, self-optimisation, etc, are implemented as Autonomic Service Agents. They use the services and data structures of the underlying autonomic networking infrastructure. The underlying Autonomic Networking Infrastructure should itself be self-managing.

The "Basic Operating System Functions" include the "normal OS", including the network stack, security functions, etc.

Full AN nodes have the full Autonomic Networking Infrastructure, with the full functionality described in this document. At a later stage ANIMA may define a scope for constrained nodes with a reduced ANI and well-defined minimal functionality. They are currently out of scope.

4. The Autonomic Networking Infrastructure

The Autonomic Networking Infrastructure provides a layer of common functionality across an Autonomic Network. It comprises "must implement" functions and services, as well as extensions.

An Autonomic Function, comprising of Autonomic Service Agents on nodes, can rely on the fact that all nodes in the network implement at least the "must implement" functions.

4.1. Naming

4.1.1. Naming requirements

- Representing each device

  Inside a domain, each autonomic device needs a domain specific identifier.

  [Open Questions] Are there devices that don’t need names? Do ASAs need names?

- Uniqueness

  The names MUST NOT collide within one autonomic domain.

  It is acceptable that the names in different domains collide, since they could be distinguished by domains.

- Semantic Encoding

  It is RECOMMENDED that the names encode some semantics rather than meaningless strings. The semantics might be:

  - Location
  - Device type
  - Functional role
  - Ownership
  - etc.

This is for ease of management consideration that network administrators could easily recognize the device directly through the names.

- Consistency

  The devices' naming SHOULD follow the same pattern within a domain.

4.1.2. Proposed Mechanisms

- Structured Naming Pattern

  The whole name string could be divided into several fields, each of which representing a specific semantic as described above. For example: Location-DeviceType-FunctionalRole-DistinguisherNumber@NameofDomain.

  The structure should be flexible that some fields are optional. When these optional fields are added, the name could still be recognized as the previous one. In above example, the "DistinguisherNumber" and "NameofDomain" are mandatory whereas others are optional. At initial stage, the devices might be only capable of self-generating the mandatory fields and the "DeviceType" because of the lack of knowledge. Later, they might have learned the "Location" and "FunctionalRole" and added the fields into current name. However, the other devices could still recognize it according to the same "DistinguisherNumber".

- Advertised Common Fields

  Some fields in the structured name might be common among the domain (e.g. "Location" "NameofDomain"). Thus, these part of the names could be advertised through Intent DistributionSection 4.5.

- Self-generated Fields

  The mandatory fields SHOULD be self-generated so that one device could name itself sufficiently without any advertised knowledges.

  There should various methods for a device to extract/generate a proper word for each mandatory semantic fields (e.g. "DeviceType", "DistinguisherNum") from its self-knowledge.

Detailed design of specific naming patterns and methods are out of scope of this document.

4.2. Addressing

Autonomic Service Agents (ASAs) need to communicate with each other, using the autonomic addressing of the node they reside on. This section describes the addressing approach of the Autonomic Networking Infrastructure, used by ASAs. It does NOT describe addressing approaches for the data plane of the network, which may be configured and managed in the traditional way, or negotiated as a service of an ASA. One use case for such an autonomic function is described in [I-D.jiang-auto-addr-management]. The addressing of the Autonomic Networking Infrastructure is in scope for this section, the address space they negotiate for the data plane is not.

Autonomic addressing is a function of the Autonomic Networking Infrastructure (lower part of Figure 2), specifically the Autonomic Control Plane. ASAs do not have their own addresses. They may use either API calls, or the autonomic addressing scheme of the Autonomic Networking Infrastructure.

An autonomic addressing scheme has the following requirements:

- o Zero-touch for simple networks: Simple networks should have complete self-management of addressing, and not require any central address management, tools, or address planning.
- o Low-touch for complex networks: If complex networks require operator input for autonomic address management, it should be limited to high level guidance only, expressed in Intent.
- o Flexibility: The addressing scheme must be flexible enough for nodes to be able to move around, for the network to grow, split and merge.
- o Robustness: It should be as hard as possible for an administrator to negatively affect addressing (and thus connectivity) in the autonomic context.
- o Support for virtualization: Autonomic Nodes may support Autonomic Service Agents in different virtual machines or containers. The addressing scheme should support this architecture.
- o Simplicity: To make engineering simpler, and to give the human administrator an easy way to trouble-shoot autonomic functions.
- o Scale: The proposed scheme should work in any network of any size.

- o Upgradability: The scheme must be able to support different addressing concepts in the future.

The primary use for the autonomically managed addressing described here is for the Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]). The fundamental concepts, as well as the proposed addressing scheme for the ACP is discussed in [I-D.behringer-anima-autonomic-addressing].

4.3. Discovery

Traditionally, most of the information a node requires is provided through configuration or northbound interfaces. An autonomic function should rely on such northbound interfaces minimally or not at all, and therefore it needs to discover peers and other resources in the network. This section describes various discovery functions in an autonomic network.

Discovering nodes and their properties and capabilities: A core function to establish an autonomic domain is the mutual discovery of autonomic nodes, primarily adjacent nodes and secondarily off-link peers. This may in principle either leverage existing discovery mechanisms, or use new mechanisms tailored to the autonomic context. An important point is that discovery must work in a network with no predefined topology, ideally no manual configuration of any kind, and with nodes starting up from factory condition or after any form of failure or sudden topology change.

Discovering services: Network services such as AAA should also be discovered and not configured. Service discovery is required for such tasks. An autonomic network can either leverage existing service discovery functions, or use a new approach, or a mixture.

Thus the discovery mechanism could either be fully integrated with autonomic signaling (next section) or could use an independent discovery mechanism such as DNS Service Discovery or Service Location Protocol. This choice could be made independently for each Autonomic Service Agent, although the infrastructure might require some minimal lowest common denominator (e.g., for discovering the security bootstrap mechanism, or the source of intent distribution, Section 4.5).

4.4. Signaling Between Autonomic Nodes

Autonomic nodes must communicate with each other, for example to negotiate and/or synchronize technical objectives (i.e., network parameters) of any kind and complexity. This requires some form of signaling between autonomic nodes. Autonomic nodes implementing a specific use case might choose their own signaling protocol, as long as it fits the overall security model. However, in the general case, any pair of autonomic nodes might need to communicate, so there needs to be a generic protocol for this. A prerequisite for this is that autonomic nodes can discover each other without any preconfiguration, as mentioned above. To be generic, discovery and signaling must be able to handle any sort of technical objective, including ones that require complex data structures. The document "A Generic Discovery and Negotiation Protocol for Autonomic Networking" [I-D.ietf-anima-grasp] describes more detailed requirements for discovery, negotiation and synchronization in an autonomic network. It also defines a protocol, GDNP, for this purpose, including an integrated but optional discovery protocol.

4.5. Intent Distribution

Intent is the policy language of an Autonomic Network; see Section 8.2 for general information on Intent. The distribution of Intent is also a function of the Autonomic Control Plane. It is expected that Intent will be expressed as quite complex human-readable data structures, and the distribution mechanism must be able to support that. Some Intent items will need to be flooded to most or all nodes, and other items of Intent may only be needed by a few nodes. Various methods could be used to distribute Intent across an autonomic domain. One approach is to treat it like any other technical objective needing to be synchronized across a set of nodes. In that case the autonomic signaling protocol could be used (previous section).

4.6. Routing

All autonomic nodes in a domain must be able to communicate with each other, and with autonomic nodes outside their own domain. Therefore, an Autonomic Control Plane relies on a routing function. For Autonomic Networks to be interoperable, they must all support one common routing protocol.

4.7. The Autonomic Control Plane

The totality of autonomic interactions forms the "Autonomic Control Plane". This control plane can be either implemented in the global routing table of a node, such as IGPs in today’s networks; or it can be provided as an overlay network. The document "An Autonomic Control Plane" ([I-D.ietf-anima-autonomic-control-plane]) describes the details.

5. Functional Overview

This section provides an overview on how the functions in the Autonomic Networking Infrastructure work together, and how the various documents about AN relate to each other.

The foundations of Autonomic Networking, definitions and gap analysis in the context of the IETF are described in [RFC7575] and [RFC7576].

Autonomic Networking is based on direct interactions between devices of a domain. The Autonomic Networking Infrastructure (ANI) is normally built on a hop-by-hop basis. Therefore, many interactions in the ANI are based on the ANI adjacency table. There are interactions that provide input into the adjacency table, and other interactions that leverage the information contained in it.

The ANI adjacency table contains information about adjacent autonomic nodes, at a minimum: node-ID, IP address in data plane, IP address in ACP, domain, certificate. An autonomic node maintains this adjacency table up to date. The adjacency table only contains information about other nodes that are capable of Autonomic Networking; non-autonomic nodes are normally not tracked here. However, the information is tracked independently of the status of the peer nodes; specifically, it contains information about non-enrolled nodes, nodes of the same and other domains. The adjacency table MAY contain information about the validity and trust of the adjacent autonomic node’s certificate, although all autonomic interactions must verify validity and trust independently.

The adjacency table is fed by the following inputs:

- Link local discovery: This interaction happens in the data plane, using IPv6 link local addressing only, because this addressing type is itself autonomic. This way the nodes learns about all autonomic nodes around itself. This is described in [I-D.ietf-anima-grasp].

- Vendor re-direct: A new device may receive information on where its home network is through a vendor based MASA re-direct; this is typically a routable address. See [I-D.pritikin-bootstrapping-keyinfrastructures].

- Non-autonomic input: A node may be configured manually with an autonomic peer; it could learn about autonomic nodes through DHCP options, DNS, and other non-autonomic mechanisms. Generally such non-autonomic mechansims require some administrator intervention. The key purpose is to by-pass a non-autonomic device or network.

As this pertains to new devices, it is covered in Section 5.3 of [I-D.pritikin-bootstrapping-keyinfrastructures].

The adjacency table is defining the behaviour of an autonomic node:

- o If the node has not bootstrapped into a domain (i.e., doesn't have a domain certificate), it rotates through all nodes in the adjacency table that claim to have a domain, and will attempt bootstrapping through them, one by one. One possible response is a vendor MASA re-direct, which will be entered into the adjacency table (see second bullet above). See [I-D.pritikin-bootstrapping-keyinfrastructures].

- o If the node has bootstrapped into a domain (i.e., has a domain certificate), it will act as a proxy for neighboring nodes that need to be bootstrapped. See [I-D.pritikin-bootstrapping-keyinfrastructures].

- o If the adjacent node has the same domain, it will authenticate that adjacent node and establish the Autonomic Control Plane (ACP). See [I-D.ietf-anima-autonomic-control-plane].

- o Other behaviours are possible, for example establishing the ACP also with devices of a sub-domain, to other domains, etc. Those will likely be controlled by Intent. They are outside scope for the moment. Note that Intent is distributed through the ACP; therefore, a node can only adapt Intent driven behaviour once it has joined the ACP. At the moment, ANIMA does not consider providing Intent outside the ACP; this can be considered later.

Once a node has joined the ACP, it will also learn the ACP addresses of its adjacent nodes, and add them to the adjacency table, to allow for communication inside the ACP. Further interactions will now happen inside the ACP. At this moment, only negotiation / synchronization via GRASP [I-D.ietf-anima-grasp] is being defined. (Note that GRASP runs in the data plane, as an input in building the adjacency table, as well as inside the ACP.)

Autonomic Functions consist of Autonomic Service Agents (ASAs). They run logically above the AN Infrastructure, and may use the adjacency table, the ACP, negotiation and synchronization through GRASP in the ACP, Intent and other functions of the ANI. Since the ANI only provides autonomic interactions within a domain, autonomic functions can also use any other context on a node, specifically the global data plane.

6. Security and Trust Infrastructure

An Autonomic Network is self-protecting. All protocols are secure by default, without the requirement for the administrator to explicitly configure security.

Autonomic nodes have direct interactions between themselves, which must be secured. Since an autonomic network does not rely on configuration, it is not an option to configure for example pre-shared keys. A trust infrastructure such as a PKI infrastructure must be in place. This section describes the principles of this trust infrastructure.

A completely autonomic way to automatically and securely deploy such a trust infrastructure is to set up a trust anchor for the domain, and then use an approach as in the document "Bootstrapping Key Infrastructures" [I-D.pritikin-bootstrapping-keyinfrastructures].

6.1. Public Key Infrastructure

An autonomic domain uses a PKI model. The root of trust is a certification authority (CA). A registrar acts as a registration authority (RA).

A minimum implementation of an autonomic domain contains one CA, one Registrar, and network elements.

6.2. Domain Certificate

We need to define how the fields in a domain certificate are to be used. [tbc]

6.3. The MASA

Explain briefly the function, point to [I-D.pritikin-bootstrapping-keyinfrastructures]. [tbc]

6.4. Sub-Domains (*)

Explain how sub-domains are handled. (tbc)

6.5. Cross-Domain Functionality (*)

Explain how trust is handled between different domains. (tbc)

7. Autonomic Service Agents (ASA)

This section describes how autonomic services run on top of the Autonomic Networking Infrastructure.

7.1. General Description of an ASA

general concepts, such as sitting on top of the ANI, etc. Also needs to explain that on a constrained node not all ASAs may run, so we have two classes of ASAs: Ones that run on an unconstrained node, and limited function ASAs that run also on constrained nodes. We expect unconstrained nodes to support all ASAs.

7.2. Specific ASAs for the Enrolment Process

The following ASAs provide essential, required functionality in an autonomic network, and are therefore mandatory to implement on unconstrained autonomic nodes.

7.2.1. The Enrolment ASA

This section describes the function of an autonomic node to bootstrap into the domain with the help of an enrolment proxy (see previous section). [tbc]

7.2.2. The Enrolment Proxy ASA

This section describes the function of an autonomic node that helps a non-enrolled, adjacent devices to enrol into the domain. [tbc]

7.2.3. The Registrar ASA

This section describes the registrar function in an autonomic network. It explains the tasks of a registrar element, and how registrars are placed in a network, redundancy between several, etc. [tbc]

8. Management and Programmability

This section describes how an Autonomic Network is managed, and programmed.

8.1. How an AN Network Is Managed

Autonomic management usually co-exists with traditional management methods in most networks. Thus, autonomic behavior will be defined for individual functions in most environments. In fact, the co-existence is twofold: autonomic functions can use traditional methods and protocols (e.g., SNMP and NETCONF) to perform management tasks; and autonomic functions can conflict with behavior enforced by the same traditional methods and protocols.

The autonomic intent is defined at a high level of abstraction. However, since it is necessary to address individual managed elements, autonomic management needs to communicate in lower-level interactions (e.g., commands and requests). For example, it is expected that the configuration of such elements be performed using NETCONF and YANG modules as well as the monitoring be executed through SNMP and MIBs.

Conflict can occur between autonomic default behavior, autonomic intent, traditional management methods. Conflict resolution is achieved in autonomic management through prioritization [RFC7575]. The rationale is that manual and node-based management have a higher priority over autonomic management. Thus, the autonomic default behavior has the lowest priority, then comes the autonomic Intent (medium priority), and, finally, the highest priority is taken by node-specific network management methods, such as the use of command line interfaces [RFC7575].

8.2. Intent (*)

This section describes Intent, and how it is managed. Intent and Policy-Based Network Management (PBNM) is already described inside the IETF (e.g., PCIM and SUPA) and in other SDOs (e.g., DMTF and TMF ZOOM).

Intent can be describe as an abstract, declarative, high-level policy used to operate an autonomic domain, such as an enterprise network [RFC7575]. Intent should be limited to high level guidance only, thus it does not directly define a policy for every network element separately. In an ideal autonomic domain, only one intent provided by human administrators is necessary to operate such domain [RFC7576]. However, it is als expected intent definition from autonomic function(s) and even from traditional network management elements (e.g., OSS).

Intent can be refined to lower level policies using different approaches, such as Policy Continuum model [ref]. This is expected in order to adapt the intent to the capabilities of managed devices. In this context, intent may contain role or function information, which can be translated to specific nodes [RFC7575]. One of the possible refinements of the intent is the refinement to Event Condition Action (ECA) rules. Such rules, which are more suitable to individual entities, can be defined using different syntax and semantics.

Different parameters may be configured for intents. These parameters are usually provided by the human operator. Some of these parameters can influence the behavior of specific autonomic functions as well as the way the intent is used to manage the autonomic domain (towards intended operational point).

Some examples of parameters for intents are:

- o Model version: The version of the model used to define the intent.

- o Domain: The network scope in which the intent has effect.

- o Name: The name of the intent which describes the intent for human operators.

- o Version: The version of the intent, which is primarly used to control intent updates.

- o Signature: The signature is used as a security mechanism to provide authentication, integrity, and non-repudiation.

- o Timestamp: The timestamp of the creation of the intent using the format supported by the IETF [TBC].

- o Lifetime: The lifetime in which the intent may be observed. A special case of the lifetime is the definition of permanent intents.

Intent distribution is considered as one of the common control and management functions of an autonomic network [RFC7575]. Since distribution is fundamental for autonomic networking, it is necessary a mechanism to provision intent by all devices in a domain [I-D.ietf-anima-grasp]. The distribution of Intent is function of the Autonomic Control Plane and several methods can be used to distribute Intent across an autonomic domain [draft-behringer-anima-reference-model]. Intent distribution might not use the ANIMA signaling protocol itself [I-D.ietf-anima-grasp], but there is a proposal to extend such protocol for intent delivery [draft-liu-anima-intent-distribution].

8.3. Aggregated Reporting (*)

Autonomic Network should minimize the need for human intervention. In terms of how the network should behave, this is done through an autonomic intent provided by the human administrator. In an analogous manner, the reports which describe the operational status of the network should aggregate the information produced in different network elements in order to present the effectiveness of autonomic intent enforcement. Therefore, reporting in an autonomic network should happen on a network-wide basis [RFC7575]. The information gathering and the reporting delivery should be done through the autonomic control plane.

Several events can occur in an autonomic network in the same way they can happen in a traditional network. These events can be produced considering traditional network management protocols, such as SNMP and syslog. However, when reporting to a human administrator, such events should be aggregated in order to avoid advertisement about individual managed elements. In this context, algorithms may be used to determine what should be reported (e.g., filtering) and in which way and how different events are related to each other. Besides that, an event in an individual element can be compensated by changes in other elements in order to maintain in a network-wide level which is described in the autonomic intent.

Reporting in an autonomic network may be in the same abstraction level of the intent. In this context, the visibility on current operational status of an autonomic network can be used to switch to different management modes. Despite the fact that autonomic management should minimize the need for user intervention, possibly there are some events that need to be addressed by human administrator actions. An alternative to model this is the use of exception-based management [RFC7575].

8.4. Feedback Loops to NOC(*)

Feedback loops are required in an autonomic network to allow the intervention of a human administrator or central control systems, while maintaining a default behaviour. Through a feedback loop an administrator can be prompted with a default action, and has the possibility to acknowledge or override the proposed default action.

8.5. Control Loops (*)

Control loops are used in autonomic networking to provide a generic mechanism to enable the Autonomic System to adapt (on its own) to various factors that can change the goals that the Autonomic System is trying to achieve, or how those goals are achieved. For example, as user needs, business goals, and the ANI itself changes, self-adaptation enables the ANI to change the services and resources it makes available to adapt to these changes.

Control loops operate to continuously observe and collect data that enables the autonomic management system to understand changes to the behavior of the system being managed, and then provide actions to move the state of the system being managed toward a common goal.

Self-adaptive systems move decision-making from static, pre-defined commands to dynamic processes computed at runtime.

Most autonomic systems use a closed control loop with feedback. Such control loops SHOULD be able to be dynamically changed at runtime to adapt to changing user needs, business goals, and changes in the ANI.

The document [draft-strassner-anima-control-loop] defines the requirements for an autonomic control loop, describes different types of control loops, and explains how control loops are used in an autonomic system.

8.6. APIs (*)

Most APIs are static, meaning that they are pre-defined and represent an invariant mechanism for operating with data. An Autonomic Network SHOULD be able to use dynamic APIs in addition to static APIs.

A dynamic API is one that retrieves data using a generic mechanism, and then enables the client to navigate the retrieved data and operate on it. Such APIs typically use introspection and/or reflection. Introspection enables software to examine the type and properties of an object at runtime, while reflection enables a program to manipulate the attributes, methods, and/or metadata of an object.

APIs MUST be able to express and preserve semantics across different domains. For example, software contracts [Meyer97] are based on the principle that a software-intensive system, such as an Autonomic Network, is a set of communicating components whose interaction is based on precisely-defined specifications of the mutual obligations that interacting components must respect. This typically includes specifying:

- o pre-conditions that MUST be satisfied before the method can start execution
- o post-conditions that MUST be satisfied when the method has finished execution
- o invariant attributes that MUST NOT change during the execution of the method

8.7. Data Model (*)

The following definitions are taken from [supa-model]:

An information model is a representation of concepts of interest to an environment in a form that is independent of data repository, data definition language, query language, implementation language, and protocol. In contrast, a data model is a representation of concepts of interest to an environment in a form that is dependent on data repository, data definition language, query language, implementation language, and protocol (typically, but not necessarily, all three).

The utility of an information model is to define objects and their relationships in a technology-neutral manner. This forms a consensual vocabulary that the ANI and ASAs can use. A data model is then a technology-specific mapping of all or part of the information model to be used by all or part of the system.

A system may have multiple data models. Operational Support Systems, for example, typically have multiple types of repositories, such as SQL and NoSQL, to take advantage of the different properties of each. If multiple data models are required by an Autonomic System, then an information model SHOULD be used to ensure that the concepts of each data model can be related to each other without technological bias.

A data model is essential for certain types of functions, such as a MRACL. More generally, a data model can be used to define the objects, attributes, methods, and relationships of a software system (e.g., the ANI, an autonomic node, or an ASA). A data model can be used to help design an API, as well as any language used to interface to the Autonomic Network.

9. Coordination Between Autonomic Functions (*)

9.1. The Coordination Problem (*)

Different autonomic functions may conflict in setting certain parameters. For example, an energy efficiency function may want to shut down a redundant link, while a load balancing function would not want that to happen. The administrator must be able to understand and resolve such interactions, to steer autonomic network performance to a given (intended) operational point.

Several interaction types may exist among autonomic functions, for example:

- Cooperation: An autonomic function can improve the behavior or performance of another autonomic function, such as a traffic forecasting function used by a traffic allocation function.

- Dependency: An autonomic function cannot work without another one being present or accessible in the autonomic network.

- Conflict: A metric value conflict is a conflict where one metric is influenced by parameters of different autonomic functions. A parameter value conflict is a conflict where one parameter is modified by different autonomic functions.

Solving the coordination problem beyond one-by-one cases can rapidly become intractable for large networks. Specifying a common functional block on coordination is a first step to address the problem in a systemic way. The coordination life-cycle consists in three states:

- At build-time, a "static interaction map" can be constructed on the relationship of functions and attributes. This map can be used to (pre-)define policies and priorities on identified conflicts.

- At deploy-time, autonomic functions are not yet active/acting on the network. A "dynamic interaction map" is created for each instance of each autonomic functions and on a per resource basis, including the actions performed and their relationships. This map provides the basis to identify conflicts that will happen at run-time, categorize them and plan for the appropriate coordination strategies/mechanisms.

- At run-time, when conflicts happen, arbitration is driven by the coordination strategies. Also new dependencies can be observed and inferred, resulting in an update of the dynamic interaction map and adaptation of the coordination strategies and mechanisms.

Multiple coordination strategies and mechanisms exists and can be devised. The set ranges from basic approaches such as random process or token-based process, to approaches based on time separation and hierarchical optimization, to more complex approaches such as multi-objective optimization, and other control theory approaches and algorithms family.

9.2. A Coordination Functional Block (*)

A common coordination functional block is a desirable component of the ANIMA reference model. It provides a means to ensure network properties and predictable performance or behavior such as stability, and convergence, in the presence of several interacting autonomic functions.

A common coordination function requires:

- A common description of autonomic functions, their attributes and life-cycle.

- o A common representation of information and knowledge (e.g., interaction maps).

- o A common "control/command" interface between the coordination "agent" and the autonomic functions.

Guidelines, recommendations or BCPs can also be provided for aspects pertaining to the coordination strategies and mechanisms.

10. Security Considerations

10.1. Threat Analysis

This is a preliminary outline of a threat analysis, to be expanded and made more specific as the various Autonomic Networking specifications evolve.

Since AN will hand over responsibility for network configuration from humans or centrally established management systems to fully distributed devices, the threat environment is also fully distributed. On the one hand, that means there is no single point of failure to act as an attractive target for bad actors. On the other hand, it means that potentially a single misbehaving autonomic device could launch a widespread attack, by misusing the distributed AN mechanisms. For example, a resource exhaustion attack could be launched by a single device requesting large amounts of that resource from all its peers, on behalf of a non-existent traffic load. Alternatively it could simply send false information to its peers, for example by announcing resource exhaustion when this was not the case. If security properties are managed autonomically, a misbehaving device could attempt a distributed attack by requesting all its peers to reduce security protections in some way. In general, since autonomic devices run without supervision, almost any kind of undesirable management action could in theory be attempted by a misbehaving device.

If it is possible for an unauthorised device to act as an autonomic device, or for a malicious third party to inject messages appearing to come from an autonomic device, all these same risks would apply.

If AN messages can be observed by a third party, they might reveal valuable information about network configuration, security precautions in use, individual users, and their traffic patterns. If encrypted, AN messages might still reveal some information via traffic analysis, but this would be quite limited (for example, this would be highly unlikely to reveal any specific information about user traffic). AN messages are liable to be exposed to third parties on any unprotected Layer 2 link, and to insider attacks even on protected Layer 2 links.

11. IANA Considerations

This document requests no action by IANA.

12. Acknowledgements

Many people have provided feedback and input to this document: Sheng Jiang, Roberta Maglione, Jonathan Hansford.

13. References

[I-D.behringer-anima-autonomic-addressing]
Behringer, M., "An Autonomic IPv6 Addressing Scheme", draft-behringer-anima-autonomic-addressing-02 (work in progress), October 2015.

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-01 (work in progress), October 2015.

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-01 (work in progress), October 2015.

[I-D.jiang-auto-addr-management]
Jiang, S., Carpenter, B., and Q. Qiong, "Autonomic Networking Use Case for Auto Address Management", draft-jiang-auto-addr-management-00 (work in progress), April 2014.

[I-D.pritikin-bootstrapping-keyinfrastructures]
Pritikin, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-pritikin-bootstrapping-keyinfrastructures-01 (work in progress), September 2014.

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

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

[RFC7404] Behringer, M. and E. Vyncke, "Using Only Link-Local Addressing inside an IPv6 Network", RFC 7404, DOI 10.17487/RFC7404, November 2014, <http://www.rfc-editor.org/info/rfc7404>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7576>.

Authors' Addresses

Michael H. Behringer (editor)
Cisco Systems
Building D, 45 Allee des Ormes
Mougins 06250
France

Email: mbehring@cisco.com

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

Email: brian.e.carpenter@gmail.com

Toerless Eckert
Cisco

Email: eckert@cisco.com

Laurent Ciavaglia
Alcatel Lucent
Route de Villejust
Nozay 91620
France

Email: laurent.ciavaglia@alcatel-lucent.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

Jeferson Campos Nobre
Federal University of Rio Grande do Sul
Av. Bento Goncalves, 9500
Porto Alegre 91501-970
Brazil

Email: jcnobre@inf.ufrgs.br

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: john.sc.strassner@huawei.com

ANIMA
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016
L. Ciavaglia
P. Peloso
Alcatel-Lucent
March 21, 2016

Autonomic Functions Coordination
draft-ciavaglia-anima-coordination-01.txt

Abstract

This document describes a management solution capable of avoiding conflicts between autonomic functions. The objective of such a solution is to avoid network instabilities, by insuring that the autonomic functions pursuing different goals will cooperate instead of antagonize each other. This document provides both requirements and specifications for such a solution.

Disclaimer: the version -01 of the draft has been issued to reactivate the document in order to allow discussion within the ANIMA WG about the coordination of autonomic functions.

Requirements Language

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].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The document Autonomic Networking: Definitions and Design Goals [RFC7575] explains the fundamental concepts behind Autonomic Networking, and defines the relevant terms in this space. The central concepts are Autonomic Nodes and Autonomic Functions.

An Autonomic Function is characterized by its implementing a closed control-loop, which we can summarize as successively:

1. Gathers metrics monitored by network equipments (that could be Autonomic Nodes, but not limited to)

2. Determines/computes new actions out of these inputs plus possibly some of the additional elements: e.g. contextual inputs, provided intents and gathered experience,

3. Set the computed parameters values (from the previous actions) inside the appropriate network equipments,

4. These new parameters values influence the network behavior, such that the metrics gathered by the autonomic function will evolve,

(Section 7.5 of [I-D.behringer-anima-reference-model] details more the control loops).

The Autonomic Functions are normally designed to stabilize (converge), at least when the network conditions are themselves stable. However, conflicting interactions among Autonomic Functions can create instabilities even when the network conditions have not varied.

The document A Reference Model for Autonomic Networking [I-D.behringer-anima-reference-model] describes the reference model of autonomic networks, by describing the architecture and enumerating fundamental blocks (either infrastructure pieces or enabling functionalities). One of these functionalities pertains to the concomitant execution of multiple autonomic functions in a safe way (i.e. avoiding conflicts between these different autonomic loops). Section 8 of [I-D.behringer-anima-reference-model] (Coordination between Autonomic Functions) provides a brief introduction to this functionality.

This document tackles this topic by successively:

1. Explaining why such a functionality is needed,

2. Detailing which objectives such a functionality should reach,

3. Sketching a simple behavior of this function,

4. Providing requirements on autonomic functions (a tentative list in this document version),

5. Providing some specifications items (in this preliminary version, while future versions would provide specifications),

2. Problem Statement

The need to coordinate the joint behavior of autonomic functions arises from the need to cope with conflicting situations and to provide the operator with the ability to steer autonomic network performance to a given (intended) operational point.

Several interaction types exist among autonomic functions such as cooperation, dependency, or conflict (and possibly others [TBD]).

Cooperation happens when an autonomic function can improve the behavior or performance of another autonomic function, such as a traffic forecasting function used by a traffic allocation function.

Dependency happens when an autonomic function cannot work without another one being present or accessible in the autonomic network.

Conflicts among autonomic functions emerges from direct and indirect interactions. A metric value conflict is a conflict where one metric is influenced by parameters of different autonomic functions. A parameter value conflict is a conflict where one parameter is modified by different autonomic functions. A simple example of conflicting interaction between autonomic functions is the oscillations caused by an energy-saving function (which switches-off interfaces to reduce power consumption) and a load-balancing function (which switches-on interfaces to reduce link load).

Solving the coordination problem beyond one-by-one cases can rapidly become intractable if one considers networks composed of tens, hundreds or thousands of simultaneously interacting functions. Specifying a common functional block on coordination is a first step to address the problem in a systemic way.

3. Guiding principles

A coordination function appears as an essential component of the ANIMA reference model in order to achieve better control on the performance, stability and convergence of autonomic networks.

As guiding principles, the ANIMA coordination function should:

- o Maximize the autonomic network utility, i.e. mitigate the (observed or inferred) detrimental effects of conflicting autonomic functions (Efficiency property).

- Balance the autonomic network goal(s) and autonomic functions individual goal(s) (Congruence and Coherence properties).

- Inform the autonomic network operator (being a human or a machine) with processed and aggregated "call(s) for governance" in case the goals are incompatible and no satisfactory solution can be found (i.e. compliant with the intent).

- Deviate the least possible autonomic functions from their design objective(s) and individual goal(s) (Liberality property).

- Impose minimal additional requirements on the external specifications of autonomic functions, such as the format and content of the autonomic function descriptor(s)/capabilities (Economy property).

- Not impose any requirement on the internal specifications of autonomic functions (Independence property).

- Support multiple coordination mechanism types (Plurality property).

- Enable coordination mechanisms to be plugged in at deploy- and run-time (Modularity property).

- Determine the most suitable coordination mechanism(s) to apply according to contexts (e.g. change in autonomic functions, change in intents/goals, change in coordination mechanisms available) (Dynamicity property).

- Develop a long-term vision of the autonomic functions interactions and devise the most suitable plan to address the conflicting cases, based on available coordination mechanisms and mission(s) set by the intent (Adaptivity property).

- Be able to fully or partially suspend/stop one or multiple autonomic functions, temporarily or an undetermined amount of time until the situation evolves (Authority property).

- Be able to operate equally well in a distributed or centralized manner (Distributivity property).

- Be able to cope with several thousands of simultaneous interactions (Performance and scalability property).

4. Initial sketch of a Coordination Function

For the sake of the following sections, this section is providing a rough description of the functioning of a coordination function, and how it organizes itself along the network time.

4.1. Preliminary assumptions

Autonomic functions do exist in different states corresponding to different steps in their life-cycle. The description of some of these steps is better understood by referring to the
High level view of an Autonomic Network which is depicted in Figure 1 of [I-D.behringer-anima-reference-model], which Figure is copied below:

```
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:             :       Autonomic Function 1        :                 :
: ASA 1       :      ASA 1      :      ASA 1      :          ASA 1  :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
              :                 :                 :
              :   +- - - - - - - - - - - - - - +  :
              :   :   Autonomic Function 2     :  :
              :   :  ASA 2      :      ASA 2   :  :
              :   +- - - - - - - - - - - - - - +  :
              :                 :                 :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
:                 Autonomic Networking Infrastructure               :
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
+--------+    :    +--------+   :    +--------+   :       +--------+
| Node 1 |--------| Node 2 |--------| Node 3 |----...-----| Node n |
+--------+    :    +--------+   :    +--------+   :       +--------+
```

Figure 1: High level view of an Autonomic Network

Undeployed - In this state, the Autonomic Function is a mere piece of software, which may not even be copied on any node, but which may well be the code of the Autonomic Service Agents (ASA) corresponding to this Autonomic Function.

Instantiated/Deployed - In this state the Autonomic Function is deployed, which means the ASA are available in the Nodes and gathered together into an Autonomic Function. In this state the autonomic function is bind to a scope which is the part of the network on which the autonomic function is meant to perform its duties. As a first approximation, the scope matches the Nodes which receive instructions from one of the ASA gathered in the Autonomic Function.

Running - In this state, the autonomic function is deployed and is executing its closed control loop, hence acting on network, by modifying Nodes parameters.

The above list of states is not meant to be exhaustive, and would be better expanded in a document dedicated to Autonomic Functions, nevertheless the distinctions between the three above states are unavoidable.

4.2. Algorithms for coordination

This sub-section does not intend to specify algorithms capable of achieving coordination between autonomic functions, but means to illustrate different ways of avoiding conflicts, we can briefly list the following families of algorithms:

Random token - This algorithm is insuring that each autonomic function is executing its control-loop the one after the other, the sequence is following a random pattern.

Time separation - This algorithm is insuring that each autonomic function is executing its control-loop at different rates, e.g. for 2 functions: one is running fast enough to have time to converge in between two iterations of the slower one (this algorithm requires proper settings with regards of the autonomic functions to coordinate).

Efficiency bids - In this algorithm, each autonomic function predicts which improvement its executing of its control-loop would bring, hence the coordination algorithms, picks the autonomic function promising the "best" improvement, and grants it the right to execute.

4.3. Behavior of the coordination function

This function is expected to steer the network towards a better "operating" point, by avoiding/mitigating detrimental interactions between Autonomic Functions.

The first step of such a process is the identification of these interactions and their classification in order to determine which ones have to be handled (at least the problematic ones i.e. conflicting ones).

The second step is the gathering of the identified interactions in groups that can be handled together while insuring the proper behavior of the network. This step intends to avoid handling all the interactions in one raw, but possibly to split the whole problem in smaller pieces, easier to handle.

The third step is the instantiation of coordination mechanisms well suited to handle each groups of interactions previously identified. Hence these coordination mechanisms would control the autonomic functions in order to insure a network behavior matching the intents of the network operator.

4.3.1. Times of the identification of interactions between AF

As the coordination function handles autonomic functions, its working is related to the different states of autonomic functions, namely, build-time, deploy-time and run-time. Hence the coordination function also present a life-cycle consisting in these 3 different states , in which the coordination function behaves according to the following descriptions:

At build-time, a common description of the autonomic function attributes (metrics, parameters, actions, capabilities...) allows to construct a "static interaction map" from the a-priori knowledge that can be derived/inferred from the functions attributes relationship. The static interaction map can be used as a first element by the operator (or mechanism) to (pre-)define policies and priorities as coordination strategies to manage the a-priori conflicts identified.

At deploy-time, autonomic functions are deployed on the network (i.e. installed, configured, instantiated...) but are not yet active/acting on the network. At this stage, for each instance of the autonomic functions and on a per resource basis, an inventory of the metrics monitored, of the actions performed and their relationships can be realized, resulting in a "dynamic interaction map". The dynamic interaction map provides the basis to identify conflicts that will happen at run-time, categorize them and plan for the appropriate coordination strategies/mechanisms.

At run-time, conflicts happens and arbitration is driven by the coordination strategies and available mechanisms. This is also the stage where new dependencies can be observed and inferred, ultimately resulting in update of the dynamic interaction map and possible adaptation of the coordination strategies and mechanisms.

4.3.2. Times of the coordination of AF

TBC

4.4. Conclusions

Some of the previous elements impact directly the coordination function, some other imply capacities of external elements such as Autonomic Functions and the Autonomic Control Plane. This conclusion is briefly categorizing and summarizing those:

Requirements onto the AF -

a descriptor of metrics and parameters/actions: a generic way of describing the inputs and outputs of the closed control loop, in order to identify the interactions.

a life-cycle: to match the process of the coordination (shortly stated, interaction identification and then conflict solving).

a common command interface of the autonomic functions: for the coordination to control the pace at which an autonomic function executes its control loop.

Requirements onto the ACP -

a common representation of information and knowledge: a function used to build the interactions maps.

Requirements onto the Coordination Function -

interaction identification: a function in charge of identifying interactions

interaction grouping: a function coping with grouping the previously identified interactions, in bundles that can be managed independently (for scalability concerns)

supporting various coordination mechanisms: to have the freedom of picking the most appropriate one.

interaction solving: a function capable of handling an independent bundle of interactions by controling the implied autonomic functions according to the picked algorithms.

5. External Requirements

At this stage of the document, this section is merely providing a structure of its content.

In order to achieve the aforementioned goals (detailed in section Section 3) a Coordination Functional Block should bring the following features:

a common description of autonomic functions attributes and its life-cycle.

a common command interface between the coordination "agent" and the autonomic functions.

a common representation of information and knowledge (cf. interaction maps).

Guidelines, recommendations or BCPs can also be provided for the aspects pertaining to the coordination strategies and mechanisms.

The coordination function requires a certain set of elements to work properly such as the autonomic function descriptor and the interaction map(s).

5.1. Autonomic Function Descriptor (AFD)

The Autonomic Function Descriptor (AFD) should contain the following elements:

actions, metrics, parameters, controlled resources.

5.2. control/command interface of AF

The Autonomic Function could be guided in its executing of its control-loop by the coordination mechanism. The guidance could range from preventing the executing of the control loop, to letting run on its own. In the middle of the range, coordination mechanism could restrain the actions, halt the control-loop at a given state of the execution (before enforcement).

This section can be expanded in conjunction with Section 7.5 of [I-D.behringer-anima-reference-model] details more the control loops.

5.3. Interaction/Information Maps

The Autonomic Control Plane(ACP) should be able to provide a view of the interactions between metrics in order to build the interaction maps. This functionality is needed to identify that metrics are coupled. E.g. the capacity of a link and its load ratio are intimately coupled, and to identify interactions between autonomic function, having this knowledge may prove instrumental.

6. Specifications

The coordination function can be decomposed in the following sub-functions:

interaction identification: in charge of identifying interactions

interaction grouping: coping with assigning the interactions to instances of cooperation mechanisms

interaction solving: coping with various algorithms

TBC.

7. Acknowledgements

This draft was written using the xml2rfc project.

This draft content builds upon work achieved during UniverSelf FP7 EU project.

The authors thank Dimitri Papadimitriou for his valuable comments.

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

TBC

10. References

10.1. Normative References

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

10.2. Informative References

[I-D.behringer-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Liu, B., Jeff, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-behringer-anima-reference-model-04 (work in progress), October 2015.

[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, <http://www.rfc-editor.org/info/rfc7575>.

Authors' Addresses

Laurent Ciavaglia
Alcatel-Lucent
Villarceaux
Nozay 91460
FR

Email: laurent.ciavaglia@alcatel-lucent.com

Peloso Pierre
Alcatel-Lucent
Villarceaux
Nozay 91460
FR

Email: pierre.peloso@alcatel-lucent.com

ANIMA WG   Z. Du
Internet-Draft   S. Jiang
Intended status: Informational   Huawei Technologies Co., Ltd
Expires: August 18, 2017   J. Nobre
                               Federal University of Rio Grande do Sul
                                                             L. Ciavaglia
                                                          Alcatel Lucent
                                                            M. Behringer
                                                           Cisco Systems
                                                       February 14, 2017

ANIMA Intent Policy and Format
draft-du-anima-an-intent-05

Abstract

One of the goals of autonomic networking is to simplify the management of networks by human operators. Intent Based Networking (IBN) is a possible approach to realize this goal. With IBN, the operator indicates to the network what to do (i.e. her intent) and not how to do it. In the field of Policy Based Management (PBM), the concept of intent is called a declarative policy. This document proposes a refinement of the intent concept initially defined in [RFC7575] for autonomic networks by providing a more complete definition, a life-cycle, some use cases and a tentative format of the ANIMA Intent Policy.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

One of the goals of autonomic networking is to simplify the management of networks by human operators. Intent Based Networking (IBN) is a possible approach to realize this goal. With IBN, the operator indicates to the network what to do (i.e. her intent) and not how to do it. In the field of Policy Based Management (PBM), the concept of intent is called a declarative policy. This document proposes a refinement of the intent concept initially defined in [RFC7575] for autonomic networks by providing a more complete definition, a life-cycle, some use cases and a tentative format of the ANIMA Intent Policy.

An Autonomic Network must be able to operate with minimum intervention from human operators. However, it still needs to receive some form of guidance (e.g. ANIMA Intent Policies) in order to fulfill the operator requirements.

In PBM, the Policy Continuum defines the levels at which the policies are defined (policy creation point), consumed (policy execution point) and translated (policy interpretation point). Using PBM, the operator can manage the network as a whole, and does not need to configure each individual devices in the network. The transformation of the high-level/abstract policies to the low-level device configurations is realized automatically by a set of functions usually regrouped inside a Policy Engine.

The use of policies and in particular of declarative policies assumes that the entities in the Autonomic Network receiving the ANIMA Intent Policy are capable of processing (refining and/or executing) the policy with no ambiguity. For that, the format of the ANIMA Intent Policy and the hierarchy of policy levels must be specified.

This document proposes a base format of the ANIMA Intent Policy. Application-specific extensions of the base format should be defined on a per need basis in dedicated documents.

2. Requirements Language and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

Autonomic Function: A feature or function which requires no configuration, and can derive all required information either through self-knowledge, discovery or through Intent.

Autonomic Node: A node which employs exclusively Autonomic Functions.

Legacy Node: A non-autonomic node, i.e., a node which employs some non-autonomic functions.

Autonomic Network: A network containing exclusively Autonomic Nodes. It may contain one or several Autonomic Domains.

Autonomic Domain: A collection of autonomic nodes that instantiate the same Intent.

Autonomic Service Agent: An agent implemented on an Autonomic Node which implements an Autonomic Function.

Intent: An abstract, high-level policy used to operate the network.

ANIMA Intent Policy: A declarative type of policy used in Autonomic Networks.

Configlet: Intent is interpreted on the Autonomic Node, and the results will be interpreted and stored in a local format on the Autonomic Node. This stored version is known as a "configlet".

NOC: A network operations center is the location where network monitoring and control is exercised.

3. Concept of ANIMA Intent Policy

In the scope of autonomic networking, the definition of intent can be found in [I-D.ietf-anima-reference-model], in which intent is described as "an abstract, declarative, high-level policy used to operate an autonomic domain, such as an enterprise network."

An Autonomic Network will comprise multiple ANIMA Intent Policies. Different ANIMA Intent Policies will be "interpreted" by different entities in autonomic networks, and the "level" of understanding of the intent will impact how the intent will be presented to this entity. So there should be "intermediate" mechanisms/functions that cater for the intent translation continuum across the heterogeneity (in policy capabilities) of the network entities. Also, ANIMA Intent Policies will possibly overlap and this overlapping should be managed (e.g., avoid conflicts, resolve applicable policies in context).

4. Intent Life Cycle

This section describes a top-down flow about how an ANIMA Intent Policy is derived through an autonomic network.

1. Business goals: The network owner wants the network to follow some business goals. These goals are initially not formalised in a particular way. A Domain Specific Language (DSL) is used to format these goals in a form subsequent components can interpret and process.

2. ANIMA Intent Policy (or Intent): Is the formalisation of business goals so that computer can deal with them. It is encoded as a file (or several files), and this file must be "given to the network".

3. Ingestion: The Intent file(s) get instantiated on an autonomic node. On a particular node, an intent file is "ingested". After that, it needs to be distributed.

4. Intent Distribution: Intent is flooded to all nodes in a network. Every node has a copy of the original "Intent" file(s), without modification. Each node re-distributes the original Intent files, without modification. Therefore, Intent is optional and transitive in nature. The Intent files must now be interpreted by each node. Editor's note: need to better defined meaning of "optional" and "transitive".

5. Intent splitting (on each node): Intent is split into sections, one for the ANI itself, others for specific Autonomic Functions. ASAs are notified if there is new Intent for them. Some intent sections may not apply to a particular node. Now each component of a node (ANI, all ASAs) know their respective Intent.

6. Intent Interpretation (on each node, by each function): The ANI as well as all ASAs on a node interpret their respective Intent section(s). It gets translated into a "target configuration", taking into account local state. For this translation, it may be necessary for ASAs to communicate with ASAs on other nodes, to pass on resources (IP addresses), to negotiate, etc. All such communications may be triggered by Intent, but the communications themselves are not Intent. (NB: This interpretation could also be done centrally, and the resulting configurations distributed; This is of course an option, but out the scope of ANIMA.) After interpreting Intent locally on each node, each node has target configlet to apply. Editor's note: define new terms such as "configlet"

7. Conflict Resolution with non-autonomic management (on each node): The target configlet resulting from Intent has the lowest priority; meanwhile, any other management method (CLI, NETCONF, etc.) overrides Intent.

8. Conflict Resolution between autonomic components (on each node): Each autonomic function needs to register with a "conflict resolution function" which parameters it modifies; in case of conflict, the conflict resolution function takes a decision and feeds that back to the autonomic functions. This may modify the target configlet.

9. Applying the target configlet.

10. Feedback loops to NOC: The NOC needs to know about certain conditions, such as conflicts with non-autonomic management.

Not all conflicts can be resolved automatically, so they may require NOC actions. Undesirable states (deviations from expected default behaviour) may have to be communicated too. To some extent, Intent itself can specify which conditions should trigger feedback loops to the NOC. Feedback loops may happen at other phases as well (ex: 8).

5. Use Cases for ANIMA Intent Policy

In this section, some use cases are introduced to clarify the concept of ANIMA Intent Policy. It should be noted that intent is defined per Autonomic Function, and can also be a general one related to multiple AFs.

The first example is about "arranging VM guest distribution". The autonomic network is supposed to be able to monitor the CPU/power utilization on each host machine, and control the status of each host machine (e.g. turn on/off). The operator may have an intent "there should be enough hosts to keep CPU utilization less than 70%", and also another one "there are few enough hosts powered so that electricity isn’t wasted".

These two intents can both influence the ASA responsible for controlling how many hosts are needed. The final decision is made according to multiple factors, including network environment and intents entered by the operators.

In this case, the first intent should have a higher priority than the later one. The two intents should be analyzed and coordinated to ensure the ASA act rightly.

Another example is about coordination of "load balancing" intent and "energy saving" intent. Autonomic Network of Operator A is composed of Autonomic Function Agents such as load balancing (LB_AFA) and energy saving (ES_AFA). Operator A wants to limit the proportion of links loaded over a certain threshold and thus defines an Intent to activate load balancing if the load is superior to 0.6 on more than 30% of the links.

Meanwhile, operator A wants different load balancing policies per (technology, administrative, topology) domain. Let’s consider a metropolitan network domain and a core network domain, or different LB policy for border routers than interior routers. For the metropolitan network domain, Operator A defines an Intent to minimize the link load variance. For the core network domain, Operator A applies the previously defined intent (activate load balancing if the load is superior to 0.6 on more than 30% of the links).

The intents will be distributed to the right network domain, and take effect after being interpreted and coordinated, and it is easy to change them without the need to configure every device manually.

6. Distribution of ANIMA Intent Policy

The distribution of intent can be done by using GRASP [I-D.ietf-anima-grasp] and ACP [I-D.ietf-anima-autonomic-control-plane]. The operator can issue a new intent or modify an intent through any authorized nodes in the autonomic network. After that, the intent will be flooded to all the nodes in the autonomic network. Another scenario is that when a new node joins into an autonomic domain, it may receive an intent from its neighbor.

For example, GRASP can be used to communicate version number of the intent, and meanwhile, a URL where to find it.

{Editor Notes: other distribution methods are also possible. }

7. Management of ANIMA Intent Policy

Every Autonomic Node in the Autonomic domain should own an intent with the same version. Any updating of intent will cause the change of the intent version number. To ensure all the nodes own the same intent, the nodes should be able to communicate with neighbors in the domain about the version of the intent. If its neighbor has a newer version of intent, it can request an intent update.

If the operator issues a new intent or modify intents, it will trigger a domain level updating of intent. Nodes in the Autonomic Network should be aware which domain it belongs to, and accept intent for that domain.

{Editor Notes: talk about the questions as follows. When/on which triggers are intents generated, updated? How the domain(s) are defined and recognized (if I am an AFA, how do I know I am part of domain x, y or z...?). }

8. Interpretation of ANIMA Intent Policy

After receiving an intent, the Autonomic Node should confirm whether it is acceptable, according to the domain name information, intent version, signature, and so on. If it passes the validation, an intent interpretation module will be involved to decide which ASAs will be involved in. Coordination of intents may be needed before the execution of the policies interpreted from the intent.

{Editor Notes: talk about the questions as follows. How the AFAs receive, understand and react to an intent? }

{Editor Notes: how the splitting (step 5 in the Life Cycle section) happens here can be explained more here. It would be better that an example can be introduced here.}

9. Uniform Format of the ANIMA Intent Policy

{Editor Notes: Format of Intent is FFS. It is suggested to contain the following information.}

This section proposes a uniform intent format. It uses the tag-based format.

Autonomic intent: The root tag for the Autonomic Network Intent.

Intent type: It indicates the intent type, which is associated with a specific Autonomic Function.

Autonomic domain: It indicates the domain of the Autonomic Network. It is also the scope of the Autonomic Network Intent.

Intent version: It indicates the version of the ANIMA Intent Policy. This is an important feature for synchronization.

Model version: The version of the model used to define the intent.

Name: The name of the intent which describes the intent for human operators.

Signature: The signature is used as a security mechanism to provide authentication, integrity, and non-repudiation.

Timestamp: The timestamp of the creation of the intent using the format supported by the IETF [TBC].

Lifetime: The lifetime in which the intent may be observed. A special case of the lifetime is the definition of permanent intents.

Content: It contains the main information of the intent. It may include objects, policies, goals and configuration data. The detailed contents and formats should be defined under their specific situations by documents that specifies the Autonomic Service Agent. Within the content, there may be sub_intents.

10. Security Considerations

Relevant security issues are discussed in [I-D.ietf-anima-grasp]. The ANIMA Intent Policy requires strong security environment from the start, because it would be great risk if the ANIMA Intent Policy had been maliciously tampered. The Autonomic Intent should employ a signature scheme to provide authentication, integrity, and non-repudiation.

11. IANA Considerations

This document defines one new format. The IANA is requested to establish a new assigned list for it.

12. Acknowledgements

The authors of this draft would like to thank the following persons for their valuable feedback and comments: Bing Liu, Brian Carpenter, Michael Richardson, Joel Halpern, John Strassner, and Jason Coleman.

This document was produced using the xml2rfc tool [RFC2629].

13. Change log [RFC Editor: Please remove]

draft-du-anima-an-intent-00: original version, 2015-06-11.

draft-du-anima-an-intent-01: add intent use case section, add some elements for the format section, and coauthor Jeferson Campos Nobre and Laurent Ciavaglia, 2015-07-06.

draft-du-anima-an-intent-02: add the intent concept section, and some other sections, 2015-10-14.

draft-du-anima-an-intent-03: modify the use case section, and add some other contents, 2016-03-17.

draft-du-anima-an-intent-04: modify the use case section, add the procedure section, and reorganize contents, 2016-07-08.

draft-du-anima-an-intent-05: modify the use case section, and delete some sections, 2017-02-15.

14. References

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-05 (work in progress), January 2017.

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-09 (work in progress), December 2016.

[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-02 (work in progress), July 2016.

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

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[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, <http://www.rfc-editor.org/info/rfc7575>.

Authors' Addresses

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

Email: duzongpeng@huawei.com

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

Email: jiangsheng@huawei.com

Jeferson Campos Nobre
Federal University of Rio Grande do Sul
Porto Alegre
Brazil

Email: jcnobre@inf.ufrgs.br

Laurent Ciavaglia
Alcatel Lucent
Route de Villejust
Nozay 91620
France

Email: laurent.ciavaglia@alcatel-lucent.com

Michael Behringer
Cisco Systems
Building D, 45 Allee des Ormes
Mougins 06250
France

Email: mbehring@cisco.com

ANIMA T. Eckert
Internet-Draft M. Behringer
Intended status: Informational Cisco
Expires: April 21, 2016 October 19, 2015

Using Autonomic Control Plane for Stable Connectivity of Network OAM
draft-eckert-anima-stable-connectivity-02

Abstract

OAM (Operations, Administration and Management) processes for data networks are often subject to the problem of circular dependencies when relying on network connectivity of the network to be managed for the OAM operations itself. Provisioning during device/network bring up tends to be far less easy to automate than service provisioning later on, changes in core network functions impacting reachability can not be automated either because of ongoing connectivity requirements for the OAM equipment itself, and widely used OAM protocols are not secure enough to be carried across the network without security concerns.

This document describes how to integrate OAM processes with the autonomic control plane (ACP) in Autonomic Networks (AN). to provide stable and secure connectivity for those OAM processes.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Self dependent OAM connectivity

OAM (Operations, Administration and Management) processes for data networks are often subject to the problem of circular dependencies when relying on network connectivity of the network to be managed for the OAM operations itself:

The ability to perform OAM operations on a network device requires first the execution of OAM procedures necessary to create network connectivity to that device in all intervening devices. This typically leads to sequential, 'expanding ring configuration' from a NOC. It also leads to tight dependencies between provisioning tools and security enrollment of devices. Any process that wants to enroll multiple devices along a newly deployed network topology needs to tightly interlock with the provisioning process that creates connectivity before the enrollment can move on to the next device.

When performing change operations on a network, it likewise is necessary to understand at any step of that process that there is no interruption of connectivity that could lead to removal of connectivity to remote devices. This includes especially change provisioning of routing, security and addressing policies in the network that often occur through mergers and acquisitions, the introduction of IPv6 or other mayor re-hauls in the infrastructure design.

All this circular dependencies make OAM processes complex and potentially fragile. When automation is being used, for example through provisioning systems or network controllers, this complexity extends into that automation software.

1.2. Data Communication Networks (DCNs)

In the late 1990'th and early 2000, IP networks became the method of choice to build separate OAM networks for the communications infrastructure in service providers. This concept was standardized in G.7712/Y.1703 and called "Data Communications Networks" (DCN). These where (and still are) physically separate IP(/MPLS) networks that provide access to OAM interfaces of all equipment that had to be managed, from PSTN switches over optical equipment to nowadays ethernet and IP/MPLS production network equipment.

Such DCN provide stable connectivity not subject to aforementioned problems because they are separate network entirely, so change configuration of the production IP network is done via the DCN but never affects the DCN configuration. Of course, this approach comes at a cost of buying and operating a separate network and this cost is not feasible for many networks, most notably smaller service providers, most enterprises and typical IoT networks.

1.3. Leveraging the ACP

One goal of the Autonomic Networks Autonomic Control plane (ACP) is to provide similar stable connectivity as a DCN, but without having to build a separate DCN. It is clear that such 'in-band' approach can never achieve fully the same level of separation, but the goal is to get as close to it as possible.

This solution approach has several aspects. One aspect is designing the implementation of the ACP in network devices to make it actually perform without interruption by changes in what we will call in this document the "data-plane", aka: the operator or controller configured services planes of the network equipment. This aspect is not currently covered in this document.

Another aspect is how to leverage the stable IPv6 connectivity provided by the ACP to build actual OAM solutions. This is the current scope of this document.

2. Solutions

2.1. Stable connectivity for centralized OAM operations

In the most common case, OAM operations will be performed by one or more applications running on a variety of centralized NOC systems that communicate with network devices. We describe differently advanced approaches to leverage the ACP for stable connectivity leveraging the ACP. The descriptions will show that there is a wide range of options, some of which are simple, some more complex.

Most easily we think there are three stages of interest:

- o There are simple options described first that we consider to be good starting points to operationalize the use of the ACP for stable connectivity.

- o The are more advanced intermediate options that try to establish backward compatibility with existing deployed approached such as leveraging NAT. Selection and deployment of these approaches needs to be carefully vetted to ensure that they provide positive RoI. This very much depends on the operational processes of the network operator.

- o It seems clearly feasible to build towards a long-term configuration that provides all the desired operational, zero touch and security benefits of an autonomic network, but a range of details for this still have to be worked out.

2.1.1. Simple connectivity for non-autonomic NOC application devices

In the most simple deployment case, the ACP extends all the way into the NOC via a network device that is set up to provide access into the ACP natively to non-autonomic devices. It acts as the default-router to those hosts and provides them with only IPv6 connectivity into the ACP - but no IPv4 connectivity. NOC devices with this setup need to support IPv6 but require no other modifications to leverage the ACP.

This setup is sufficient for troubleshooting OAM operations such as SSH into network devices, NMS that perform SNMP read operations for status checking, for software downloads into autonomic devices and so on. In conjunction with otherwise unmodified OAM operations via separate NOC devices/applications it can provide a good subset of the interesting stable connectivity goals from the ACP.

Because the ACP provides 'only' for IPv6 connectivity, and because the addressing provided by the ACP does not include any addressing structure that operations in a NOC often relies on to recognize where devices are on the network, it is likely highly desirable to set up DNS so that the ACP IPv6 addresses of autonomic devices are known via domain names with logical names. For example, if DNS in the network was set up with names for network devices as devicename.noc.example.com, then the ACP address of that device could be mapped to devicename-acp.noc.exmaple.com.

2.1.2. Limitations and enhancement overview

This most simple type of attachment of NOC applications to the ACP suffers from a range of limitations:

1. NOC applications can not directly probe whether the desired so called 'data-plane' network connectivity works because they do not directly have access to it. This problem is not dissimilar to probing connectivity for other services (such as VPN services) that they do not have direct access to, so the NOC may already employ appropriate mechanisms to deal with this issue (probing proxies).

2. NOC applications need to support IPv6 which often is still not the case in many enterprise networks.

3. Performance of the ACP will be limited versus normal 'data-plane' connectivity. The setup of the ACP will often support only non-hardware accelerated forwarding. Running a large amount of traffic through the ACP, especially for tasks where it is not necessary will reduce its performance/effectiveness for those operations where it is necessary or highly desirable.

4. Security of the ACP is reduced by exposing the ACP natively (and unprotected) into a LAN In the NOC where the NOC devices are attached to it.

These four problems can be tackled independently of each other by solution improvements. Combining these solutions improvements together ultimately leads towards the the target long term solution.

2.1.3. Simultaneous ACP and data plane connectivity

Simultaneous connectivity to both ACP and data-plane can be achieved in a variety of ways. If the data-plane is only IPv4, then any method for dual-stack attachment of the NOC device/application will suffice: IPv6 connectivity from the NOC provides access via the ACP, IPv4 will provide access via the data-plane. If as explained above in the most simple case, an autonomic device supports native attachment to the ACP, and the existing NOC setup is IPv4 only, then it could be sufficient to simply attach the ACP device(s) as the IPv6 default-router to the NOC LANs and keep the existing IPv4 default router setup unchanged.

If the data-plane of the network is also supporting IPv6, then the NOC devices that need access to the ACP should have a dual-homing IPv6 setup. One option is to make the NOC devices multi-homed with one logical or physical IPv6 interface connecting to the data-plane, and another into the ACP. The LAN that provides access to the ACP should then be given an IPv6 prefix that shares a common prefix with the IPv6 ULA of the ACP so that the standard IPv6 interface selection rules on the NOC host would result in the desired automatic selection of the right interface: towards the ACP facing interface for connections to ACP addresses, and towards the data-plane interface for anything else. If this can not be achieved automatically, then it needs to be done via simple IPv6 static routes in the NOC host.

Providing two virtual (eg: dot1q subnet) connections into NOC hosts may be seen as undesired complexity. In that case the routing policy to provide access to both ACP and data-plane via IPv6 needs to happen in the NOC network itself: The NOC application device gets a single attachment interface but still with the same two IPv6 addresses as in before - one for use towards the ACP, one towards the data-plane. The first-hop router connecting to the NOC application device would then have separate interfaces: one towards the data-plane, one towards the ACP. Routing of traffic from NOC application hosts would then have to be based on the source IPv6 address of the host: Traffic from the address designated for ACP use would get routed towards the ACP, traffic from the designated data-plane address towards the data-plane.

In the most simple case, we get the following topology: Existing NOC application devices connect via an existing NOClan and existing first hop Rtr1 to the data-plane. Rtr1 is not made autonomic, but instead the edge router of the Autonomic network ANrtr is attached via a separate interface to Rtr1 and ANrtr provides access to the ACP via ACPaccessLan. Rtr1 is configured with the above described IPv6 source routing policies and the NOC-app-devices are given the secondary IPv6 address for connectivity into the ACP.

```
                                   --... (data-plane)
NOC-app-device(s) -- NOClan -- Rtr1
                                   --- ACPaccessLan -- ANrtr ... (ACP)
```

Figure 1

If Rtr1 was to be upgraded to also implement Autonomic Networking and the ACP, the picture would change as follows:

```
                                           ---- ... (data-plane)
      NOC-app-device(s) ---- NOClan --- ANrtr1
                                       .  .   ---- ... (ACP)
                                       \-/
                                       (ACP to data-plane loopback)
```

Figure 2

In this case, ANrtr1 would have to implement some more advanced routing such as cross-VRF routing because the data-plane and ACP are most likely run via separate VRFs. A simple short-term workaround could be a physical external loopback cable into two ports of ANrtr1 to connect the data-plane and ACP VRF as shown in the picture.

2.1.4. IPv4 only NOC application devices

With the ACP being intentionally IPv6 only, attachment of IPv4 only NOC application devices to the ACP requires the use of IPv4 to IPv6 NAT. This NAT setup could for example be done in Rt1r1 in above picture to also support IPv4 only NOC application devices connected to NOClan.

To support connections initiated from IPv4 only NOC applications towards the ACP of network devices, it is necessary to create a static mapping of ACP IPv6 addresses into an unused IPv4 address space and dynamic or static mapping of the IPv4 NOC application device address (prefix) into IPv6 routed in the ACP. The main issue in this setup is the mapping of all ACP IPv6 addresses to IPv4. Without further network intelligence, this needs to be a 1:1 address mapping because the prefix used for ACP IPv6 addresses is too long to be mapped directly into IPv4 on a prefix basis.

One could implement in router software dynamic mappings by leveraging DNS, but it seems highly undesirable to implement such complex technologies for something that ultimately is a temporary problem (IPv4 only NOC application devices). With today's operational directions it is likely more preferable to automate the setup of 1:1 NAT mappings in that NAT router as part of the automation process of network device enrollment into the ACP.

The ACP can also be used for connections initiated by the network device into the NOC application devices. For example syslog from autonomic devices. In this case, static mappings of the NOC application devices IPv4 addresses are required. This can easily be done with a static prefix mapping into IPv6.

Overall, the use of NAT is especially subject to the RoI considerations, but the methods described here may not be too different from the same problems encountered totally independent of AN/ACP when some parts of the network are to introduce IPv6 but NOC application devices are not (yet) upgradeable.

2.1.5. Path selection policies

As mentioned above, the ACP is not expected to have high performance because its primary goal is connectivity and security, and for existing networ device platforms this often means that it is a lot more effort to implement that additional connectivity with hardware acceleration than without - especially because of the desire to support full encryption across the ACP to achieve the desired security.

Some of these issues may go away in the future with further adoption of the ACP and network device designs that better tender to the needs of a separate OAM plane, but it is wise to plan for even long-term designs of the solution that does NOT depend on high-performance of the ACP. This is opposite to the expectation that future NOC application devices will have IPv6, so that any considerations for IPv4/NAT in this solution are temporary.

To solve the expected performance limitations of the ACP, we do expect to have the above describe dual-connectivity via both ACP and data-plane between NOC application devices and AN devices with ACP. The ACP connectivity is expected to always be there (as soon as a device is enrolled), but the data-plane connectivity is only present under normal operations but will not be present during eg: early stages of device bootstrap, failures, provisioning mistakes or during network configuration changes.

The desired policy is therefore as follows: In the absence of further security considerations (see below), traffic between NOC application and AN devices should prefer data-plane connectivity and resort only to using the ACP when necessary, unless it is an operation known to be so much tied to the cases where the ACP is necessary that it makes no sense to try using the data plane. An example here is of course the SSH connection from the NOC into a network device to troubleshoot network connectivity. This could easily always rely on the ACP. Likewise, if a NOC application is known to transmit large amounts of data, and it uses the ACP, then its performance need to be controlled so that it will not overload the ACP performance. Typical examples of this are software downloads.

There is a wide range of methods to build up these policies. We describe a few:

DNS can be used to set up names for the same network devices but with different addresses assigned: One name (name.noc.example.com) with only the data-plane address(es) (IPv4 and/or IPv6) to be used for probing connectivity or performing routine software downloads that may stall/fail when there are connectivity issues. One name (name-acp.noc.example.com) with only the ACP reachable address of the device for troubleshooting and probing/discovery that is desired to always only use the ACP. One name with data plane and ACP addresses (name-both.noc.example.com).

Traffic policing and/or shaping of at the ACP edge in the NOC can be used to throttle applications such as software download into the ACP.

MP-TCP is a very attractive candidate to automate the use of both data-plane and ACP and minimize or fully avoid the need for the above mentioned logical names to pre-set the desired connectivity (data-plane-only, ACP only, both). For example, a set-up for non MP-TCP aware applications would be as follows:

DNS naming is set up to provide the ACP IPv6 address of network devices. Unbeknownst to the application, MP-TCP is used. MP-TCP mutually discovers between the NOC and network device the data-plane address and caries all traffic across it when that MP-TCP sub-flow across the data-plane can be built.

In the Autonomic network devices where data-plane and ACP are in separate VRFs, it is clear that this type of MP-TCP sub-flow creation across different VRFs is new/added functionality. Likewise the policies of preferring a particular address (NOC-device) or VRF (AN device) for the traffic is potentially also a policy not provided as a standard.

2.1.6. Autonomic NOC device/applications

Setting up connectivity between the NOC and autonomic devices when the NOC device itself is non-autonomic is as mentioned in the beginning a security issue. It also results as shown in the previous paragraphs in a range of connectivity considerations, some of which may be quite undesirable or complex to operationalize.

Making NOC application devices autonomic and having them participate in the ACP is therefore not only a highly desirable solution to the security issues, but can also provide a likely easier operationalization of the ACP because it minimizes NOC-special edge considerations - the ACP is simply built all the way automatically, even inside the NOC and only authorized and authenticate NOC devices/ applications will have access to it.

Supporting the ACP all the way into an application device requires implementing the following aspects in it: AN bootstrap/enrollment mechanisms, the secure channel for the ACP and at least the host side of IPv6 routing setup for the ACP. Minimally this could all be implemented as an application and be made available to the host OS via eg: a tap driver to make the ACP show up as another IPv6 enabled interface.

Having said this: If the structure of NOC applications is transformed through virtualization anyhow, then it may be considered equally secure and appropriate to construct a (physical) NOC application system by combining a virtual AN/ACP enabled router with non-AN/ACP enabled NOC-application VMs via a hypervisor, leveraging the configuration options described in the previous sections but jut virtualizing them.

2.1.7. Encryption of data-plane connections

When combining ACP and data-plane connectivity for availability and performance reasons, this too has an impact on security: When using the ACP, the traffic will be mostly encryption protected, especially when considering the above described use of AN application devices. If instead the data-plane is used, then this is not the case anymore unless it is done by the application.

The most simple solution for this problem exists when using AN NOC application devices, because in that case the communicating AN NOC application and the AN network device have certificates through the AN enrollment process that they can mutually trust (same AN domain). In result, data-plane connectivity that does support this can simply leverage TLS/dTLS with mutual AN-domain certificate authentication - and does not incur new key management.

If this automatic security benefit is seen as most important, but a "full" ACP stack into the NOC application device is unfeasible, then it would still be possible to design a stripped down version of AN functionality for such NOC hosts that only provides enrollment of the NOC host into the AN domain to the extend that the host receives an AN domain certificate, but without directly participating in the ACP afterwards. Instead, the host would just leverage TLS/dTLS using its AN certificate via the data-plane with AN network devices as well as indirectly via the ACP with the above mentioned in-NOC network edge connectivity into the ACP.

When using the ACP itself, TLS/dTLS for the transport layer between NOC application and network device is somewhat of a double price to pay (ACP also encrypts) and could potentially be optimized away, but given the assumed lower performance of the ACP, it seems that this is an unnecessary optimization.

2.1.8. Long term direction of the solution

If we consider what potentially could be the most lightweight and autonomic long term solution based on the technologies described above, we see the following direction:

1. NOC applications should at least support IPv6. IPv4/IPv6 NAT in the network to enable use of ACP is long term undesirable. Having IPv4 only applications automatically leverage IPv6 connectivity via host-stack options is likely non-feasible (NOTE: this has still to be vetted more).

2. Build the ACP as a lightweight application for NOC application devices so ACP extends all the way into the actual NOC application devices.

3. Leverage and as necessary enhance MP-TCP with automatic dual-connectivity: If the MP-TCP unaware application is using ACP connectivity, the policies used should add sub-flow(s) via the data-plane and prefer them.

4. Consider how to best map NOC application desires to underlying transport mechanisms: With the above mentioned 3 points, not all options are covered. Depending on the OAM operation, one may still want only ACP, only data-plane, or automatically prefer one over the other and/or use the ACP with low performance or high-performance (for emergency OAM actions such as countering DDoS). It is as of today not clear what the simplest set of tools is to enable explicitly the choice of desired behavior of each OAM operations. The use of the above mentioned DNS and MP-TCP

mechanisms is a start, but this will require additional thoughts. This is likely a specific case of the more generic scope of TAPS.

2.2. Stable connectivity for distributed network/OAM functions

The ACP can provide common direct-neighbor discovery and capability negotiation and stable and secure connectivity for functions running distributed in network devices. It can therefore eliminate the need to re-implement similar functions in each distributed function in the network. Today, every distributed protocol does this with functional elements usually called "Hello" mechanisms and with often protocol specific security mechanisms.

KARP has tried to start provide common directions and therefore reduce the re-invention of at least some of the security aspects, but it only covers routing-protocols and it is unclear how well it applicable to a potentially wider range of network distributed agents such as those performing distributed OAM functions. The ACP can help in these cases.

This section is TBD for further iterations of this draft.

3. Security Considerations

We discuss only security considerations not covered in the appropriate sub-sections of the solutions described.

Even though ACPs are meant to be isolated, explicit operator misconfiguration to connect to insecure OAM equipment and/or bugs in ACP devices may cause leakage into places where it is not expected. Mergers/Aquisitions and other complex network reconfigurations affecting the NOC are typical examples.

ULA addressing as proposed in this document is preferred over globally reachable addresses because it is not routed in the global Internet and will therefore be subject to more filtering even in places where specific ULA addresses are being used.

Randomn ULA addressing provides more than sufficient protection against address collision even though there is no central assignment authority. This is helped by the expectation, that ACPs are never expected to connect all together, but only few ACPs may ever need to connect together, eg: when mergers and aquisitions occur.

If packets with unexpected ULA addresses are seen and one expects them to be from another networks ACP from which they leaked, then some form of ULA prefix registrastion (not allocation) can be beneficial. Some voluntary registries exist, for example https://www.sixxs.net/tools/grh/ula/, although none of them is preferrable because of being operated by some recognized authority. If an operator would want to make its ULA prefix known, it might need to register it with multiple existing registries.

ULA Centrally assigned ULA addresses (ULA-C) was an attempt to introduce centralized registration of randomnly assigned addresses and potentially even carve out a different ULA prefix for such addresses. This proposal is currently not proceeding, and it is questionable whether the stable connectivity use case provides sufficient motivation to revive this effort.

Using current registration options implies that there will not be reverse DNS mapping for ACP addresses. For that one will have to rely on looking up the unknown/unexpected network prefix in the registry registry to determine the owner of these addresses.

Reverse DNS resolution may be beneficial for specific already deployed insecure legacy protocols on NOC OAM systems that intend to communicate via the ACP (eg: TFTP) and leverages reverse-DNS for authentication. Given how the ACP provides path security except potentially for the last-hop in the NOC, the ACP does make it easier to extend the lifespan of such protocols in a secure fashion as far to just the transport is concerned. The ACP does not make reverse DNS lookup a secure authentication method though. Any current and future protocols must rely on secure end-to-end communications (TLD, dTLS) and identification and authentication via the certificates assigned to both ends. This is enabled by the certificate mechanisms of the ACP.

If DNS and especially reverse DNS are set up, then it should be set up in an automated fashion, linked to the autonomic registrar backend so that the DNS and reverse DNS records are actually derived from the subject name elements of the ACP device certificates in the same way as the autonomic devices themselves will derive their ULA addresses from their certificates to ensure correct and consistent DNS entries.

If an operator feels that reverse DNS records are beneficial to its own operations but that they should not be made available publically for "security" by concealment reasons, then the case of ACP DNS entries is probably one of the least problematic use cases for split-DNS: The ACP DNS names are only needed for the NOC applications intending to use the ACP - but not network wide across the enterprise.

4. No IPv4 for ACP

The ACP is targeted to be IPv6 only, and the prior explanations in this document show that this can lead to some complexity when having to connect IPv4 only NOC solutions, and that it will be impossible to leverage the ACP when the OAM agents on an ACP network device do not support IPv6. Therefore, the question was raised whether the ACP should optionally also support IPv4.

The decision not to include IPv4 for ACP as something that is considered in the use cases in this document is because of the following reasons:

In SP networks that have started to support IPv6, often the next planned step is to consider moving out IPv4 from a native transport as just a service on the edge. There is no benefit/need for multiple parallel transport families within the network, and standardizing on one reduces OPEX and improves reliability. This evolution in the data plane makes it highly unlikely that investing development cycles into IPv4 support for ACP will have a longer term benefit or enough critical short-term use-cases. Support for only IPv4 for ACP is purely a strategic choice to focus on the known important long term goals.

In other type of networks as well, we think that efforts to support autonomic networking is better spent in ensuring that one address family will be support so all use cases will long-term work with it, instead of duplicating effort into IPv4. Especially because auto-addressing for the ACP with IPv4 would be more ecomplex than in IPv6 due to the the IPv4 addressing space.

5. Further considerations

6. IANA Considerations

This document requests no action by IANA.

7. Acknowledgements

This work originated from an Autonomic Networking project at cisco Systems, which started in early 2010 including customers involved in the design and early testing. Many people contributed to the aspects described in this document, including in alphabetical order: BL Balaji, Steinthor Bjarnason, Yves Herthoghs, Sebastian Meissner, Ravi Kumar Vadapalli. The author would also like to thank Michael Richardson, James Woodyatt and Brian Carpenter for their review and comments.

8. Change log [RFC Editor: Please remove]

02: Updated references.

02: Modified ULA text to not suggest ULA-C as much better anymore, but still mention it.

02: Added explanation why no IPv4 for ACP.

01: Added security section discussing the role of address prefix selection and DNS for ACP. Title change to emphasize focus on OAM. Expanded abstract.

00: Initial version.

9. References

[I-D.behringer-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Liu, B., Jeff, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-behringer-anima-reference-model-04 (work in progress), October 2015.

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-01 (work in progress), October 2015.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", draft-ietf-anima-bootstrapping-keyinfra-01 (work in progress), October 2015.

[I-D.irtf-nmrg-an-gap-analysis]
Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-06 (work in progress), April 2015.

[I-D.irtf-nmrg-autonomic-network-definitions]
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking - Definitions and Design Goals", draft-irtf-nmrg-autonomic-network-definitions-07 (work in progress), March 2015.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

Authors' Addresses

Toerless Eckert
Cisco

Email: eckert@cisco.com

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

ANIMA WG T. Eckert, Ed.
Internet-Draft Futurewei USA
Intended status: Standards Track M. Behringer, Ed.
Expires: May 5, 2020

S. Bjarnason
Arbor Networks
November 2, 2019

An Autonomic Control Plane (ACP)
draft-ietf-anima-autonomic-control-plane-21

Abstract

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

Table of Contents

1. Introduction (Informative)

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

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

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

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

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

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

The Autonomic Control Plane serves several purposes at the same time:

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

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

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

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

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

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

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

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

Combining ACP with Bootstrapping Remote Secure Key Infrastructures (BRSKI), see [I-D.ietf-anima-bootstrapping-keyinfra]) results in the "Autonomic Network Infrastructure" as defined in [I-D.ietf-anima-reference-model], which provides autonomic connectivity (from ACP) with 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 and confidentiality mechanism used by the ACP is defined to be negotiable, therefore it can be extended to environments with different protocol preferences. The minimum implementation requirements in this document attempt to achieve maximum interoperability by requiring support for multiple options depending on the type of device: IPsec, see [RFC4301], and datagram Transport Layer Security version 1.2 (DTLS), see [RFC6347]).

The implementation footprint of the ACP consists of Public Key Infrastructure (PKI) code for the ACP certificate, the GRASP protocol, UDP, TCP and TLS (for security and reliability of GRASP), the ACP secure channel protocol used (such as IPsec or DTLS), and an instance of IPv6 packet forwarding and routing via the Routing Protocol for Low-power and Lossy Networks (RPL), see [RFC6550], that is separate from routing and forwarding for the Data-Plane (user traffic).

The ACP uses only IPv6 to avoid complexity of dual-stack ACP operations (IPv6/IPv4). Nevertheless, it can without any changes be integrated into even otherwise IPv4-only network devices. The Data-Plane itself would not need to change, it could continue to be IPv4 only. For such IPv4 only devices, the IPv6 protocol itself would be additional implementation footprint only used for the ACP.

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

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

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

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

2. Acronyms and Terminology (Informative)

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

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

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

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

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

ACP address: An IPv6 address assigned to the ACP node. It is stored in the domain information field of the ->"ACP domain certificate" ().

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

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

ACP domain: The ACP domain is the set of nodes with ->"ACP domain certificates" that allow them to authenticate each other as members of the ACP domain. See also Section 6.1.3.

ACP (ANI/AN) Domain Certificate: A [RFC5280] certificate (LDevID) carrying the domain information field which is used by the ACP to learn its address in the ACP and to derive and cryptographically assert its membership in the ACP domain.

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

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

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

ACP (ULA) prefix(es): The /48 IPv6 address prefixes used across the ACP. In the normal/simple case, the ACP has one ULA prefix, see Section 6.10. The ACP routing table may include multiple ULA prefixes if the "rsub" option is used to create addresses from more than one ULA prefix. See Section 6.1.2. 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 cryptographically authenticated and encrypted data connection 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 6.12.5.

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

(AN) Domain Name: An FQDN (Fully Qualified Domain Name) in the domain information field of the Domain Certificate. See Section 6.1.2.

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 products of the IETF ANIMA working group.

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

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

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

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

device: A physical system, or physical node.

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

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

GRASP: "Generic Autonomic Signaling Protocol". An extensible signaling protocol required by the ACP for ACP neighbor discovery.

The ACP also provides the "security and transport substrate" for the "ACP instance of GRASP". This instance of GRASP runs across the ACP secure channels to support BRSKI and other NOC/OAM or Autonomic Functions. See [I-D.ietf-anima-grasp].

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

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

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

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

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

MIC: "Manufacturer Installed Certificate". 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.

node: A system, e.g., supporting the ACP according to this document. Can be virtual or physical. Physical nodes are called devices.

Node-ID: The identifier of an ACP node inside that ACP. It is the last 64 (see Section 6.10.3) or 78-bits (see Section 6.10.5) of the ACP address.

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

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

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

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

(ACP/ANI/BRSKI) Registrar: An ACP registrar is an entity (software and/or person) that is orchestrating the enrollment of ACP nodes with the ACP domain certificate. ANI nodes use BRSKI, so ANI registrars are also called BRSKI registrars. For non-ANI ACP nodes, the registrar mechanisms are undefined by this document. See Section 6.10.7. Renewal and other maintenance (such as revocation) of ACP domain certificates may be performed by other entities than registrars. EST must be supported for ACP domain certificate renewal (see Section 6.1.5). BRSKI is an extension of EST, so ANI/BRSKI registrars can easily support ACP domain certificate renewal in addition to initial enrollment.

sUDI: "secured Unique Device Identifier". This is another word to describe an IDevID in referenced material. This term is not used in this document.

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

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

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

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

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

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

3.1. An Infrastructure for Autonomic Functions

Autonomic Functions need a stable infrastructure to run on, and all autonomic functions should use the same infrastructure to minimize the complexity of the network. In this way, there is only need for a single discovery mechanism, a single security mechanism, and single instances of other processes that distributed functions require.

3.2. Secure Bootstrap over a not configured Network

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

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

3.3. Data-Plane Independent Permanent Reachability

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

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

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

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

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

- o 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.

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

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

4. Requirements (Informative)

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

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

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

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

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

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

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

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

5. Overview (Informative)

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

1. An ACP node creates a Virtual Routing and Forwarding (VRF) instance, or a similar virtual context.

2. It determines, following a policy, a candidate peer list. This is the list of nodes to which it should establish an Autonomic Control Plane. Default policy is: To all link-layer adjacent nodes supporting ACP.

3. For each node in the candidate peer list, it authenticates that node (according to Section 6.1.3) and negotiates a mutually acceptable channel type.

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

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

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

Note:

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

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

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

The following figure illustrates the ACP.

```
             ACP node 1                          ACP node 2
          ...................               ...................
   secure .                 .   secure      .                 .  secure
   channel:  +-----------+  :   channel     :  +-----------+  : channel
   ..--------| ACP VRF   |---------------------| ACP VRF   |---------..
          : / \         / \   <--routing-->   / \         / \ :
          : \ /         \ /                   \ /         \ / :
   ..--------| Loopback  |---------------------| Loopback  |---------..
          :  | interface |  :               :  | interface |  :
          :  +-----------+  :               :  +-----------+  :
          :                 :               :                 :
          :   Data-Plane    :...............:   Data-Plane    :
          :                 :    link       :                 :
          :.................:               :.................:
```

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

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

6.1. ACP Domain, Certificate and Network

The ACP relies on group security. An ACP domain is a group of nodes that trust each other to participate in ACP operations. To establish trust, each ACP member requires keying material: An ACP node MUST have a certificate (LDevID) and a Trust Anchor (TA) consisting of a certificate (chain) used to sign the LDevID of all ACP domain members. The LDevID is used to cryptographically authenticate the membership of its owner node in the ACP domain to other ACP domain members. The TA is used to authenticate the ACP domain membership of other nodes (see Section 6.1.3).

The LDevID is called the ACP domain certificate, the TA is the Certificate Authority (CA) of the ACP domain.

The ACP does not mandate specific mechanisms by which this keying material is provisioned into the ACP node. It only requires the certificate to comply with Section 6.1.1, specifically the Domain information field as specified in Section 6.1.2 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 domain certificate is that domain certificate when ACP nodes are (fully) autonomic nodes. Finally, this document uses the term ACP network to refer to the network created by active ACP nodes in an ACP domain. The ACP network itself can extend beyond ACP nodes through the mechanisms described in Section 8.1.

6.1.1. ACP Certificates

ACP domain certificates MUST be [RFC5280] compliant X.509 certificates.

ACP nodes MUST support RSA and Elliptic Curve Diffie-Hellman (ECDH) public keys in ACP certificates. ACP nodes MUST support RSA and Elliptic Curve Diffie-Hellman (ECDH) signatures for ACP 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 ECDH key and an ECDH signature if the ACP certificate is only used for ACP and ANI authentication and authorization. ACP nodes MUST support 2048-bit RSA using SHA-256, SHA-384 or SHA-512 or Elliptic Curve using NIST P-256, P-384, or P-521 as key lengths in ACP certificates/signatures.

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 [RFC4492] for more information.

The reason for these choices are as follows: As of 2019, RSA is more widely used than ECDH, therefore the MUST for RSA. ECDH offers equivalent security at shorter key lengths (see [RFC4492]). 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 authenticatio/authorization, therefore ECDH can only reliably be used in the ACP when it MUST be supported on all ACP nodes.

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

The ACP domain certificate SHOULD be used for any authentication between nodes with ACP domain certificates (ACP nodes and NOC nodes) where the 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 6.1.3 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 6.6) and for ACP GRASP (Section 6.8.2) end-to-end via TLS ([RFC5246]).

The ACP domain membership check requires a minimum amount of elements in a certificate as described in Section 6.1.3. All elements are [RFC5280] compliant. The identity of a node in the ACP is carried via the ACP Domain Information Field as defined in Section 6.1.2 which is encoded as an rfc822Name field.

Any other field of the ACP domain certificate is to be populated as required by [RFC5280] or desired by the operator of the ACP domain ACP registrars/CA and required by other purposes that the ACP domain certificate is intended to be used for.

For diagnostic and other operational purposes, it is beneficial to copy the device identifying fields of the node's IDevID into the ACP domain certificate, such as the "serialNumber" (see [I-D.ietf-anima-bootstrapping-keyinfra] section 2.3.1). Inclusion of this information makes it easier to potentially attack the node though, for example by learning the device model, which may help to select a fitting subset of attacks. Neighboring attackers can retrieve the certificate through an otherwise unsuccessful initiation of a secure channel association.

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 future requirements. Such future authorizations can use and require additional elements in certificates or policies or even additional certificates. For an example, see Appendix A.10.5.

6.1.2. ACP Certificate ACP Domain Information Field

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

[RFC Editor: Please substitute SELF in all occurrences of rfcSELF in this document with the RFC number assigned to this document and remove this comment line]

```
domain-information = local-part "@" acp-domain-name
local-part = key [ "." local-info ]
key = "rfcSELF"
local-info = [ acp-address ] [ "+" rsub extensions ]
acp-address = 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 = ; future standard definition.
            ; Must fit RFC5322 simple dot-atom format.

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 extenstion of area51.research

then this results in:
domain-information = rfcSELF+fd89b714f3db00000200000064000000
                     +area51.research@acp.example.com
acp-domain-name    = acp.example.com
routing-subdomain  = area51.research.acp.example.com
```

Figure 2: ACP Domain Information Field ABNF

domain-information is the encoded information that is put into the ACP domain certificates subjectAltName / rfc822Name field. routing-subdomain is a string that can be constructed from the domain-information, and it is used in the hash-creation of the ULA (see below). The requirements and sementics of the parts of this information are explained in the following paragraphs:

Nodes complying with this specification MUST be able to receive their ACP address through the domain certificate, in which case their own ACP domain certificate MUST have the 32HEXDIG "acp-address" field. Nodes complying with this specification MUST also be able to authenticate nodes as ACP domain members / ACP secure channel peers when they have an empty or 0-value acp-address field. See Section 6.1.3.

"acp-domain-name" is used to indicate the ACP Domain across which all ACP nodes trust each other and are willing to build ACP channels to each other. See Section 6.1.3. Acp-domain-name SHOULD be the FQDN of a DNS domain owned by the operator assigning the certificate. This is a simple method to ensure that the domain is globally unique and collision of ACP addresses would therefore only happen due to ULA hash collisions (see Section 6.10.2). If the operator does not own any FQDN, it should choose a string (in FQDN format) that it intends to be equally unique.

To keep the encoding simple, there is no consideration for internationalized acp-domain-names. The ACP domain information is not intended for enduser consumption, and there is no protection against someone not owning a domain name to simpy choose it. Instead, it 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 heuristic that allows a Registrar to consistently generate a unique 48-bit ULA prefix for ACP addresses. The presence of the "rsub" component allows to a single ACP domain to employ multiple /48 ULA prefixes. See Appendix A.7 for example use-cases.

The optional "extensions" field is used for future standardized extensions to this specification. It MUST be ignored if present and not understood.

Formatting notes:

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

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

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

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

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

o It should be possible to use the ACP domain certificate as an LDevID 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.

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

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

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

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

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

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

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

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

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

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

6.1.3. ACP domain membership check

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

1: The peer certificate is valid (lifetime).

2: 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.

3: The peer's certificate passes certificate path validation as defined in [RFC5280], section 6 against one of the Trust Anchors associated with the ACP node's ACP domain certificate (see Section 6.1.4 below).

4: If the node certificate indicates a Certificate Revocation List (CRL) Distribution Point (CDP) ([RFC5280], section 4.2.1.13) or Online Certificate Status Protocol (OCSP) responder ([RFC5280], section 4.2.2.1), then the peer's certificate must be valid according to those criteria: An OCSP check for the peer's certificate across the ACP must succeed or the peer certificate must not be listed in the CRL retrieved from the CDP. This rule has to be skipped for ACP secure channel peer authentication when the node has no ACP or non-ACP connectivity to retrieve current

CRL or access an OCSP responder (and the security association protocol itself has also no way to communicate CRL or OCSP check).

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

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

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

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

Technically, ACP secure channels can only be built with nodes that have an acp-address. Rule 6 ensures that this is taken into account during ACP domain membership check.

Nodes with an empty 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 domain certificate. These ACP nodes are permitted to establish ACP secure channels. Mechanisms for those nodes to determine their ACP address are outside the scope of this specification, but this option is defined here so that any ACP nodes can build ACP secure channels to them according to Rule 6.

In summary:

Steps 1...4 constitute standard certificate validity verification and private key authentication as defined by [RFC5280] and security association protocols (such as 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. Steps 5 and 6 are the equivalent steps.

Step 4 Constitutes standard CRL/OCSP checks refined for the case of missing connectivity and limited functionality security association protocols.

Step 5 Checks for the presence of an ACP identity for the peer.

Steps 1...5 authorize to build any secure connection between members of the same ACP domain except for ACP secure channels.

Step 6 is the additional verification of the presence of an ACP address.

Steps 1...6 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.

6.1.4. Trust Points and Trust Anchors

ACP nodes need Trust Point (TP) certificates to perform certificate path validation as required by Section 6.1.3, rule 3. Trust Point(s) must be provisioned to an ACP node (together with its ACP domain certificate) by an ACP Registrar during initial enrolment of a candidate ACP node. ACP nodes MUST also support renewal of TPs via EST as described below in Section 6.1.5.

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

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

Note that different ACP nodes may have different Trust Points and even different Trust Anchors in their certificate path, as long as the set of Trust Points for all ACP node includes the same set of

Trust Anchors (usually 1), and each ACP nodes set of Trust Anchors includes the intermediate Trust Points for its own ACP domain certificate. The protocols through which ACP domain membership check rules 1-4 are performed therefore need to support the exchange not only of the ACP nodes certificates, but also their intermediate Trust Points.

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

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

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

6.1.5. Certificate and Trust Point Maintenance

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

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

The EST server MUST present a certificate that is passing ACP domain membership check in its TLS connection setup (Section 6.1.3, rules 1..5, not rule 6 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 extension 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.

6.1.5.1. 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', TCP, 443]
]
```

Figure 3: GRASP SRV.est example

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

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]
                             ; 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 4: GRASP SRV.est definition

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

6.1.5.2. 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.

6.1.5.3. Certificate Revocation Lists (CRLs)

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

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

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

6.1.5.4. 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 domain certificates are managed via a CA chain where the assigning CA has enough performance to manage short lived certificates. See also Section 10.2.4 for discussion about an example setup achieving this. See also [I-D.ietf-acme-star].

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

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

6.1.5.5. Re-enrollment

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

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

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

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

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

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

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

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

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

6.1.5.6. Failing Certificates

An ACP domain 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 domain certificate as the culprit, the peer should pass the domain membership check (Section 6.1.3) 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 domain certificate, such as for the TLS connection to an EST server for the renewal of the ACP domain certificate.

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

ACP nodes SHOULD support the option to determines whether its ACP certificate is failing, and when it does, put itself into the role of a re-enrolling candidate ACP node as explained above (Section 6.1.5.5).

6.2. ACP Adjacency Table

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

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

The adjacency table MAY contain information about the validity and trust of the adjacent ACP node's certificate. However, subsequent steps MUST always start with the ACP domain membership check against the peer (see Section 6.1.3).

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.

6.3. Neighbor Discovery with DULL GRASP

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

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

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

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

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

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

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

Figure 5: GRASP AN_ACP example

The formal CDDL definition is:

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

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

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

Figure 6: GRASP AN_ACP definition

The objective-flags field is set to indicate synchronization.

The loop-count is fixed at 1 since this is a link-local operation.

In the above example the RECOMMENDED period of sending of the objective is 60 seconds. The indicated ttl of 210000 msec means that the objective would be cached by ACP nodes even when two out of three messages are dropped in transit.

The session-id is a random number used for loop prevention (distinguishing a message from a prior instance of the same message). In DULL this field is irrelevant but must still be set according to the GRASP specification.

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

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

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

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

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

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

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

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

Note that a node serving both as an ACP node and BRSKI Join Proxy may choose to distribute the "AN_ACP" objective and the respective BRSKI in the same M_FLOOD message, since GRASP allows multiple objectives in one message. This may be impractical though if ACP and BRSKI operations are implemented via separate software modules / ASAs.

The result of the discovery is the IPv6 link-local address of the neighbor as well as its supported secure channel protocols (and non-standard port they are running on). It is stored in the ACP Adjacency Table (see Section 6.2), which then drives the further building of the ACP to that neighbor.

Note that the DULL GRASP objective described does intentionally not include ACP nodes ACP domain certificate even though this would be useful for diagnostics and to simplify the security exchange in ACP secure channel security association protocols (see Section 6.7). 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.

6.4. Candidate ACP Neighbor Selection

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

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

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

6.5. Channel Selection

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

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

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

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

- o 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.

- o Once the first secure channel protocol succeeds, the two peers know each other's certificates because they must be used by all secure channel protocols for mutual authentication. The node with the lower Node-ID in the ACP address of its ACP domain certificate becomes Bob, the one with the higher Node-ID in the certificate Alice. A peer with an empty ACP address field in its ACP domain certificate becomes Bob (this specification does not define such peers, only the interoperability with them).

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

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

The following sequence of steps show this example in more detail:

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

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

[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 Bob and Node 2 Alice
       on connection C1. Connection setup C1 is completed.

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

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

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

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

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

Figure 7: Secure Channel sequence of steps

All this negotiation is in the context of an "L2 interface". Alice and Bob will build ACP connections to each other on every "L2 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.

6.6. Candidate ACP Neighbor verification

Independent of the security association protocol chosen, candidate ACP neighbors need to be authenticated based on their domain certificate. This implies that any secure channel protocol MUST support certificate based authentication that can support the ACP domain membership check as defined in Section 6.1.3. 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 a minimum delay of 10 seconds and a maximum delay of 640 seconds.

6.7. Security Association (Secure Channel) protocols

Due to Channel Selection (Section 6.5), ACP can support an evolving set of security association protocols. These protocols only need to be used to establish secure channels with L2 adjacent ACP neighbors and only optionally (where needed) across non-ACP capable L3 network (see Section 8.2). Therefore, there is architecturally no need for any network wide mandatory to implement (MTI) security association protocols. Instead, ACP nodes only need to implement those protocols required to be supported by their neighbors. See Section 6.7.3 for an example of this.

The authentication of peers in any security association protocol MUST use the ACP domain certificate according to Section 6.1.3. Because auto-discovery of candidate ACP neighbors via GRASP (see Section 6.3) as specified in this document does not communicate the neighbors ACP domain 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.

Any security association protocol MUST use PFS (such as profiles providing PFS).

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. This principle does not imply that the requirements against ACP security association protocols have to be the same across all subnets in an ACP network:

- o Underlying 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.

- o 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 (Figure 15) 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.

The following sub-sections define the security association protocols that are considered to be important and feasible to specify in this document.

6.7.1. ACP via IKEv2

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

6.7.1.1. Native IPsec

To run ACP via IPsec natively, no further IANA assignments/ definitions are required.

An ACP node that is supporting native IPsec MUST use IPsec security setup via IKEv2 for tunnel mode and IPsec/IKE signaling accordingly for IPv6 payload (e.g.: ESP next header of 41). It MUST use local and peer link-local IPv6 addresses for encapsulation.

Authentication MUST use the ACP domain certificates. Certificate Encoding MUST support "PKCS #7 wrapped X.509 certificate" (0) (see [IKEV2IANA] for this and other IANA IKEv2 parameter names used in this text). If certificate chains are used, all intermediate certificates up to, but not including the locally provisioned trust anchor certificate must be signaled.

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

IPsec MUST support ESP with ENCR_AES_GCM_16 ([RFC4106]) due to its higher performance over ENCR_AES_CBC. ACP MUST NOT use any NULL encryption option due to the confidentiality of ACP payload that may not be encrypted by itself (when carrying legacy management protocol traffics as well as hop-by-hop GRASP).

These IPsec requirements are based on [RFC8221] but limited to the minimum necessary options because ACP is not a general purpose use case today with a wide range of interoperability requirements against legacy devices originally developed against older profile recommendations. Once there are updates to [RFC8221], these should accordingly be reflected in updates to these ACP requirements (for example if ENCR_AES_GCM_16 was to be superceeded in the future).

Additional requirements from [RFC8221] MAY be used for ACP channels as long as they do not result in a reduction of security over the above MTI requirements. For example, ESP compression MAY be used.

IKEv2 MUST follow [RFC8247] as necessary to support the above listed IPsec requirements.

6.7.1.2. IPsec with GRE encapsulation

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

To run ACP via GRE/IPsec, no further IANA assignments/definitions are required.

The requirements for ESP/IPsec/IKEv2 are the same as for native IPsec (see Section 6.7.1.1) except that IPsec transport mode and next protocol GRE (47) are to be negotiated. Tunnel mode is not required because of GRE.

If IKEv2 initiator and responder support IPsec over GRE, it has to be preferred over native IPsec. The ACP IPv6 traffic has to be carried across GRE according to [RFC7676].

6.7.2. ACP via DTLS

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

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

All ACP nodes supporting DTLS as a secure channel protocol MUST adhere to the DTLS implementation recommendations and security considerations of BCP 195 [RFC7525] except with respect to the DTLS version. ACP nodes supporting DTLS MUST implement only DTLS 1.2 or later. For example, implementing DTLS-1.3 ([I-D.ietf-tls-dtls13]) is also an option.

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

6.7.3. ACP Secure Channel Requirements

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

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

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

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

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

6.8. GRASP in the ACP

6.8.1. GRASP as a core service of the ACP

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

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

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

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

6.8.2. ACP as the Security and Transport substrate for GRASP

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

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

In this case ASAs using GRASP running on the same node would still need to be able to discover each other's objectives. When the ACP does not exist, ASAs leveraging the ACP instance of GRASP via APIs MUST still be able to operate, and MUST be able to understand that there is no ACP and that therefore the ACP instance of GRASP cannot operate.

The way ACP acts as the security and transport substrate for GRASP is visualized in the following picture:

```
..............................ACP..............................
.                                                             .
.         /-GRASP-flooding-\         ACP GRASP instance       .
.        /                  \                                 A
.    GRASP      GRASP      GRASP                              C
.  link-local   unicast  link-local                           P
.   multicast  messages   multicast                           .
.   messages      |       messages                            .
.      |          |          |                                .
...............................................................
.      v          v          v    ACP security and transport  .
.      |          |          |    substrate for GRASP         .
.      |          |          |                                .
.      |       ACP GRASP     |       - ACP GRASP              A
.      |       Loopback      |         Loopback interface     C
.      |       interface     |       - ACP-cert auth          P
.      |         TLS         |                                .
.   ACP GRASP     |       ACP GRASP  - ACP GRASP virtual      .
.   subnet1       |       subnet2      virtual interfaces     .
.     TCP         |         TCP                               .
.      |          |          |                                .
...............................................................
.      |          |          |   ^^^ Users of ACP (GRASP/ASA) .
.      |          |          |   ACP interfaces/addressing    .
.      |          |          |                                .
.      |          |          |                                A
.      | ACP-Loopback Interf.|      <- ACP Loopback interface C
.      |      ACP-address    |       - address (global ULA)   P
.    subnet1      |        subnet2  <- ACP virtual interfaces .
.  link-local     |      link-local  - link-local addresses   .
...............................................................
.      |          |          |   ACP VRF                      .
.      |     RPL-routing     | virtual routing and forwarding .
.      |   /IP-Forwarding\   |                                A
.      |  /               \  |                                C
.  ACP IPv6 packets   ACP IPv6 packets                        P
.      |/                   \|                                .
.    IPsec/DTLS        IPsec/DTLS  - ACP-cert auth            .
...............................................................
         |                   |   Data-Plane
         |                   |
         |                   |     - ACP secure channel
     link-local        link-local  - encapsulation addresses
       subnet1            subnet2  - Data-Plane interfaces
         |                   |
      ACP-Nbr1            ACP-Nbr2
```

Figure 8: ACP as security and transport substrate for GRASP

GRASP unicast messages inside the ACP always use the ACP address. Link-local addresses from the ACP VRF must not be used inside objectives. GRASP unicast messages inside the ACP are transported via TLS according to [RFC7525] execept that only TLS version 1.2 ([RFC5246]) or higher MUST be used - because there is no need for backward compatibility in the new use-case of ACP. Mutual authentication MUST use the ACP domain membership check defined in (Section 6.1.3).

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

TLS for GRASP 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 256bit AES or less than SHA384. TLS for GRASP MUST also include the "Supported Elliptic Curves" extension, it MUST support support the NIST P-256 (secp256r1) and P-384 (secp384r1(24)) curves [RFC4492]. In addition, GRASP TLS clients SHOULD send an ec_point_formats extension with a single element, "uncompressed". For further interoperability recommendations, GRASP TLS implementations SHOULD follow [RFC7525].

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

6.8.2.1. Discussion

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

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

If GRASP peer connections 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:

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

6.9. Context Separation

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

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

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

6.10. Addressing inside the ACP

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

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

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

6.10.1. Fundamental Concepts of Autonomic Addressing

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

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

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

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

- No external connectivity: They do not provide access to the Internet. If a node requires further reaching connectivity, it should use another, traditionally managed address scheme in parallel.

- Addresses in the ACP are permanent, and do not support temporary addresses as defined in [RFC4941].

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

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

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

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

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

6.10.2. The ACP Addressing Base Scheme

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

```
  8      40                     2                     78
+--+-------------------------+------+------------------------------+
|fd| hash(routing-subdomain) | Type |      (sub-scheme)            |
+--+-------------------------+------+------------------------------+
```

Figure 9: ACP Addressing Base Scheme

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

- "fd" identifies a locally defined ULA address.

- The 40-bits ULA "global ID" (term from [RFC4193]) for ACP addresses carried in the domain information field of domain certificates are the first 40-bits of the SHA256 hash of the routing subdomain from the same domain information field. In the example of Section 6.1.2, 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 others.

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

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

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

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

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

The following summarizes the addressing schemes:

type	Z	name	F-bit	V-bit size
0x00	0	ACP Zone	N/A	1 bit
0x00	1	ACP Manual	N/A	1 bit
0x01	N/A	VLong-ASA	0	8-bits
0x01	N/A	VLong-ACP-edge	1	16-bits

Figure 10: Addressing schemes

6.10.3. ACP Zone Addressing Sub-Scheme

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

```
                 64                             64
+-----------------+---+---------++-----------------------------+---+
|  (base scheme)  | Z | Zone-ID ||           Node-ID             |
|                 |   |         || Registrar-ID |   Node-Number| V |
+-----------------+---+---------++--------------+--------------+---+
         50         1     13            48             15        1
```

Figure 11: ACP Zone Addressing Sub-Scheme

The fields are defined as follows:

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

o Z: MUST be 0.

o Node-ID: A unique value for each node.

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

o Registrar-ID (48-bit): A number unique inside the domain that identifies the ACP registrar which assigned the Node-ID to the node. A MAC address of the ACP registrar can be used for this purpose.

o Node-Number: A number which is unique for a given ACP registrar, to identify the node. This can be a sequentially assigned number.

o V (1-bit): Virtualization bit: 0: Indicates the ACP itself ("ACP node base system); 1: Indicates the optional "host" context on the ACP node (see below).

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

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

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

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

6.10.3.1. Usage of the Zone-ID Field

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

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

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

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

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

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

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

6.10.4. ACP Manual Addressing Sub-Scheme

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

```
               64                             64
+---------------------+---+----------++-----------------------------+
|    (base scheme)    | Z | Subnet-ID||     Interface Identifier    |
+---------------------+---+----------++-----------------------------+
         50             1    13
```

Figure 12: ACP Manual Addressing Sub-Scheme

The fields are defined as follows:

o Subnet-ID: Configured subnet identifier.

o Z: MUST be 1.

o Interface Identifier.

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

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

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

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

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

6.10.5. ACP Vlong Addressing Sub-Scheme

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

```
          50                               78
+---------------------++-----------------------------+----------+
|     (base scheme)   ||              Node-ID                   |
|                     || Registrar-ID |F| Node-Number|        V |
+---------------------++--------------+--------------+----------+
          50                  46        1   23/15         8/16
```

Figure 13: 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 6.1.2, the V-bits are always set to 0.

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

- 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.

6.10.6. Other ACP Addressing Sub-Schemes

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

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

6.10.7. ACP Registrars

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

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

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

6.10.7.1. Use of BRSKI or other Mechanism/Protocols

Any protocols or mechanisms may be used as ACP registrars, as long as the resulting ACP certificate and trust anchors allow to perform the ACP domain membership described in Section 6.1.3 with other ACP domain members, and meet the ACP addressing requirements for its ACP domain information field as described further below in this section.

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

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

6.10.7.2. Unique Address/Prefix allocation

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

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

When the candidate ACP device (called Pledge in BRSKI) is to be enrolled into an ACP domain, the ACP registrar needs to allocate a unique ACP address to the node and ensure that the ACP certificate gets a domain information field (Section 6.1.2) with the appropriate information - ACP domain-name, ACP-address, and so on. If the ACP registrar uses BRSKI, it signals the ACP domain information field to the Pledge via the EST /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]

6.10.7.3. Addressing Sub-Scheme Policies

The ACP registrar selects for the candidate ACP node a unique address prefix from an appropriate ACP addressing sub-scheme, either a zone addressing sub-scheme prefix (see Section 6.10.3), or a Vlong addressing sub-scheme prefix (see Section 6.10.5). The assigned ACP address prefix encoded in the domain information field of the ACP domain certificate indicates to the ACP node its ACP address information. The sub-addressing scheme indicates the prefix length: /127 for zone address sub-scheme, /120 or /112 for Vlong address sub-scheme. The first address of the prefix is the ACP address. All other addresses in the prefix are for other uses by the ACP node as described in the zone and Vlong addressing sub scheme sections. The ACP address prefix itself is then signaled by the ACP node into the ACP routing protocol (see Section 6.11) to establish IPv6 reachability across the ACP.

The choice of addressing sub-scheme and prefix-length in the Vlong address sub-scheme is subject to ACP registrar policy. It could be an ACP domain wide policy, or a per ACP node or per ACP node type policy. For example, in BRSKI, the ACP registrar is aware of the IDevID of the candidate ACP node, which contains a "serialNnumber" that is typically indicating the 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. Allocation and use of zone sub-addresses with Zone-ID != 0 is outside the scope of this specification because it would need to go along with rules for extending ACP routing to multiple zones, which is outside the scope of this specification.

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

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

6.10.7.4. Address/Prefix Persistence

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

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

ACP address information SHOULD also be maintained even after an ACP certificate did expire or failed. See Section 6.1.5.5 and Section 6.1.5.6.

6.10.7.5. Further Details

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

6.11. Routing in the ACP

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

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

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

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

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

6.11.1. RPL Profile

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

6.11.1.1. Overview

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

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

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

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

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

The benefit of this profile, especially compared to other 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.

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

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

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

Additionally, failed ACP tunnels can be quickly discovered trough 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 imediately signal the RPL control plane to pick a different parent.

6.11.1.2. RPL Instances

Single RPL instance. Default RPLInstanceID = 0.

6.11.1.3. Storing vs. Non-Storing Mode

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

6.11.1.4. DAO Policy

Proactive, aggressive DAO state maintenance:

- o Use K-flag in unsolicited DAO indicating change from previous information (to require DAO-ACK).

- o Retry such DAO DAO-RETRIES(3) times with DAO- ACK_TIME_OUT(256ms) in between.

6.11.1.5. Path Metric

Hopcount.

6.11.1.6. Objective Function

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

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

6.11.1.7. DODAG Repair

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

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

stretch_rank: none provided ("not stretched").

Data Path Validation: Not used.

Trickle: Not used.

6.11.1.8. Multicast

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

6.11.1.9. Security

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

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

6.11.1.10. P2P communications

Not used.

6.11.1.11. IPv6 address configuration

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

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

6.11.1.12. Administrative parameters

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

- o Explicit configured "root": 0b100

- o ACP registrar (Default): 0b011

- o ACP-connect (non-registrar): 0b010

- o Default: 0b001.

6.11.1.13. RPL Data-Plane artifacts

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

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

6.11.1.14. Unknown Destinations

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

As this requirement raises additional data plane requirements, it does not apply to nodes where the administrative parameter to become root (Section 6.11.1.12) can always only be 0b001, e.g.: the node does not support explicit configuration to be root, or to be ACP registrar or to have ACP-connect functionality. If an ACP network is degraded to the point where there are no nodes that could be configured roots, ACP registrars or ACP-connect nodes, traffic to unknown destinations could not be diagnosed, but in the absence of any intelligent nodes supporting other than 0b001 administrative preference, there is likely also no diagnostic function possible.

6.12. General ACP Considerations

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

6.12.1. Performance

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

6.12.2. Addressing of Secure Channels

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

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

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

6.12.3. MTU

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

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

6.12.4. Multiple links between nodes

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

6.12.5. ACP interfaces

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

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

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

For these reason, the ACP (ULA) address(es) are assigned to Loopback interface(s).

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

ACP point-to-point virtual interface:

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

ACP multi-access virtual interface:

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

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

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

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

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

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

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

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

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

Assume a LAN with three ACP neighbors, Alice, Bob and Carol. Alice's ACP GRASP wants to send a link-local GRASP multicast message to Bob and Carol. If Alice's ACP emulates the LAN as one point-to-point virtual interface to Bob and one to Carol, The sending applications itself will send two copies, if Alice's ACP emulates a LAN, GRASP will send one packet and the ACP will replicate it. The result is the same. The difference happens when Bob and Carol receive their packet. If they use ACP point-to-point virtual interfaces, their GRASP instance would forward the packet from Alice to each other as part of the GRASP flooding procedure. These packets are unnecessary and would be discarded by GRASP on receipt as duplicates (by use of the GRASP Session ID). If Bob and Carol's ACP would emulate a multi-access virtual interface, then this would not happen, because GRASPs flooding procedure does not replicate back packets to the interface that they were received from.

Note that link-local GRASP multicast messages are not sent directly as IPv6 link-local multicast UDP messages into ACP virtual interfaces, but instead into ACP GRASP virtual interfaces, that are layered on top of ACP virtual interfaces to add TCP reliability to link-local multicast GRASP messages. Nevertheless, these ACP GRASP virtual interfaces perform the same replication of message and, therefore, result in the same impact on flooding. See Section 6.8.2 for more details.

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

Care must also be taken when creating multi-access ACP virtual interfaces across ACP secure channels between ACP nodes in different domains or routing subdomains. 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)

```
ANrtr1 ------ ANswitch1 --- ANswitch2 ------- ANrtr2
          .../   \                   \  ...
ANrtrM ------     \                   ------- ANrtrN
                   ANswitchM ...
```

Figure 14: 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.

If the device with L2 ports is supporting per L2 port ACP DULL GRASP as well as MLD snooping ([RFC4541]), then MLD snooping must be changed to never forward packets for ALL_GRASP_NEIGHBORS because that would cause the problem that per L2 port ACP DULL GRASP is meant to overcome (forwarding DULL GRASP packets across L2 ports).

The rest of ACP operations can operate in the same way as in L3 devices: Assume for example that the device is an L3/L2 hybrid device where L3 interfaces are assigned to VLANs and each VLAN has potentially multiple ports. DULL GRASP is run as described individually on each L2 port. When it discovers a candidate ACP neighbor, it passes its IPv6 link-local address and supported secure channel protocols to the ACP secure channel negotiation that can be bound to the L3 (VLAN) interface. It will simply use link-local IPv6 multicast packets to the candidate ACP neighbor. Once a secure channel is established to such a neighbor, the virtual interface to which this secure channel is mapped should then actually be the L2 port and not the L3 interface to best map the actual physical topology into the ACP virtual interfaces. See Section 6.12.5 for more details about how to map secure channels into ACP virtual interfaces. Note that a single L2 port can still have multiple ACP neighbors if it connect for example to multiple ACP neighbors via a non-ACP enabled switch. The per L2 port ACP virtual interface can therefore still be a multi-access virtual LAN.

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

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

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

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

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

8. Support for Non-ACP Components (Normative)

8.1. ACP Connect

8.1.1. Non-ACP Controller / NMS system

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

If the NMS host is not autonomic, i.e., it does not support autonomic negotiation of the ACP, then it can be brought into the ACP by explicit configuration. To support connections to adjacent non-ACP nodes, an ACP node must support "ACP connect" (sometimes also called "autonomic connect"):

"ACP connect" is an interface level configured workaround for connection of trusted non-ACP nodes to the ACP. The ACP node on which ACP connect is configured is called an "ACP edge node". With ACP connect, the ACP is accessible from those non-ACP nodes (such as NOC systems) on such an interface without those non-ACP nodes having to support any ACP discovery or ACP channel setup. This is also called "native" access to the ACP because to those (NOC) systems the interface looks like a normal network interface (without any encryption/novel-signaling).

```
                                 Data-Plane "native" (no ACP)
                                             .
  +--------+       +----------------+        .         +-------------+
  | ACP    |       |ACP Edge Node   |        .         |             |
  | Node   |       |                |        v         |             |
  |        |-------|...[ACP VRF]....+------------------|             |+
  |        |   ^   |.               |                  | NOC Device  ||
  |        |   .   | .[Data-Plane]..+------------------| "NMS hosts" ||
  |        |   .   |  [          ]  | .          ^     |             ||
  +--------+   .   +----------------+  .         .     +-------------+|
               .                        .        .      +-------------+
               .                        .        .
           Data-Plane "native"          .     ACP "native" (unencrypted)
         + ACP auto-negotiated          .    "ACP connect subnet"
           and encrypted                .
                                      ACP connect interface
                                      e.g., "VRF ACP native" (config)
```

Figure 15: 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 must be physically controlled/ secured. For this reason the mechanisms described here do explicitly not include options to allow for a non-ACP router to be connected across an ACP connect interface and addresses behind such a router routed inside the ACP.

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

An ACP connect interface SHOULD use an IPv6 address/prefix from the ACP Manual Addressing Sub-Scheme (Section 6.10.4), letting the operator configure for example only the Subnet-ID and having the node automatically assign the remaining part of the prefix/address. It SHOULD NOT use a prefix that is also routed outside the ACP so that the addresses clearly indicate whether it is used inside the ACP or not.

The prefix of ACP connect subnets MUST be distributed by the ACP edge node into the ACP routing protocol (RPL). The NMS hosts MUST connect to prefixes in the ACP routing table via its ACP connect interface. In the simple case where the ACP uses only one ULA prefix and all ACP connect subnets have prefixes covered by that ULA prefix, NMS hosts can rely on [RFC6724] to determine longest match prefix routes towards its different interfaces, ACP and data-plane. With RFC6724, The NMS host will select the ACP connect interface for all addresses in the ACP because any ACP destination address is longest matched by the address on the ACP connect interface. If the NMS hosts ACP connect interface uses another prefix or if the ACP uses multiple ULA prefixes, then the NMS hosts require (static) routes towards the ACP interface for these prefixes.

When an ACP Edge node receives a packet from an ACP connect interface, 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 MAY be changed through administrative measures.

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

8.1.2. Software Components

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

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

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

8.1.3. Auto Configuration

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

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

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

The ACP edge nodes that have a Vlong ACP address MAY allocate a subset of their /112 or /120 address prefix to ACP connect interface(s) to eliminate the need to non-autonomically configure/ provision the address prefixes for such ACP connect interfaces.

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

```
                        Combined ACP and Data-Plane interface
                                              .
     +--------+       +--------------------+     .   +--------------+
     | ACP    |       |ACP Edge No         |     .   | NMS Host(s)  |
     | Node   |       |                    |     .   | / Software   |
     |        |       |  [ACP  ].          |     .   |              |+
     |        |       | .[VRF  ] .[VRF   ] |     v   | "ACP address"||
     |        +-------+.         .[Select].+---------+ "Date Plane   ||
     |        |   ^   | .[Data ].          |         |  Address(es)"||
     |        |   .   |  [Plane]           |         |              ||
     |        |   .   |  [     ]           |         +--------------+|
     +--------+   .   +--------------------+          +--------------+
                  .
           Data-Plane "native" and + ACP auto-negotiated/encrypted
```

Figure 16: VRF select

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

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

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

If the ACP carries multiple ULA prefixes or non-ULA ACP connect prefixes, then the correct source address selection becomes even more problematic.

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

To achieve the same behavior with a Combined ACP and Data-Plane interface, the ACP Edge Node needs to behave as two separate routers on the interface: One link-local IPv6 address/router for its ACP reachability, and one link-local IPv6 address/router for its Data-Plane reachability. The Router Advertisements for both are as described above (Section 8.1.3): For the ACP, the ACP prefix is announced together with RFC4191 option for the prefixes routed across the ACP and lifetime=0 to disqualify this next-hop as a default router. For the Data-Plane, the Data-Plane prefix(es) are announced together with whatever dafault 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 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 of the network connection between an ACP edge node and the NMS or other host reachable via the ACP connect interface. Node integrity too is expected to be easier because the ACP connect node, the ACP connect link and the nodes connecting to it must be in a contiguous secure location, hence assuming there can be no physical attack against the devices.

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

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

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

8.2.1. Configured Remote ACP neighbor

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

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

Figure 17: 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:

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

o 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.

o 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.

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

o 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.

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

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

8.2.2. Tunneled Remote ACP Neighbor

An IPinIP, GRE or other form of pre-existing tunnel is configured between two remote ACP peers and the virtual interfaces representing the tunnel are configured for "ACP enable". This will enable IPv6 link local addresses and DULL on this tunnel. In result, the tunnel is used for normal "L2 adjacent" candidate ACP neighbor discovery with DULL and secure channel setup procedures described in this document.

Tunneled Remote ACP Neighbor requires two encapsulations: the configured tunnel and the secure channel inside of that tunnel. This makes it in general less desirable than Configured Remote ACP Neighbor. Benefits of tunnels are that it may be easier to implement because there is no change to the ACP functionality - just running it over a virtual (tunnel) interface instead of only native interfaces. The tunnel itself may also provide PMTUD while the secure channel method may not. Or the tunnel mechanism is permitted/possible through some firewall while the secure channel method may not.

8.2.3. Summary

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

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

9. Benefits (Informative)

9.1. Self-Healing Properties

The ACP is self-healing:

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

- o 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.

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

Highly resilient ACP designs can be built by using ACP registrars with embedded sub-CA, as outlined in Section 10.2.4. As long as a partition is left with one or more of such ACP registrars, it can continue to enroll new candidate ACP nodes as long as the ACP 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 trust anchor(s), a re-merge will be smooth.

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

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

9.2. Self-Protection Properties

9.2.1. From the outside

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

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

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

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

9.2.2. From the inside

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

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

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

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

9.3. The Administrator View

An ACP is self-forming, self-managing and self-protecting, therefore has minimal dependencies on the administrator of the network. Specifically, since it is (intended to be) independent of configuration, there is 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 10.2) 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.

10. ACP Operations (Informative)

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

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

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

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

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

10.1. ACP (and BRSKI) Diagnostics

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

The basic diagnostics is support of (yang) data models representing the complete (auto-)configuration and operational state of all components: BRSKI, GRASP, ACP and the infrastructure used by them: TLS/DTLS, IPsec, certificates, trust anchors, time, VRF and so on. While necessary, this is not sufficient:

Simply representing the state of components does not allow operators to quickly take action - unless they do understand how to interpret the data, and that can mean a requirement for deep understanding of all components and how they interact in the ACP/ANI.

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

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

o 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.

o 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.

o Certain operations are difficult or impossible to monitor in real-time, such as initial bootstrap issues in a network location where no capabilities exist to attach local diagnostics. Therefore it is important to also define means of capturing (logging) diagnostics locally for later retrieval. Ideally, these captures are also non-volatile so that they can survive extended power-off conditions - for example when a device that fails to be brought up zero-touch is being sent back for diagnostics at a more appropriate location.

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

Q: Is ACP operational to accept neighbor connections:

- o Check if any potentially necessary configuration to make ACP/ANI operational are correct (see Section 10.3 for a discussion of such commands).

- o Does the system time look reasonable, or could it be the default system time after clock chip battery failure (certificate checks depend on reasonable notion of time).

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

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

- o Check the validity of the domain certificate:

    * Does the certificate validate against the trust anchor?

    * Has it been revoked?

    * Was the last scheduled attempt to retrieve a CRL successful (e.g., do we know that our CRL information is up to date).

    * Is the certificate valid: validity start time in the past, expiration time in the future?

    * Does the certificate have a correctly formatted ACP domain information field?

- o Was the ACP VRF successfully created?

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

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

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

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

o Is it enabled for ACP?

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

o Do we receive DULL GRASP messages on the interface? If not, some intervening L2 equipment performing bad MLD snooping could have caused problems. Provide e.g., diagnostics of the MLD querier IPv6 and MAC address.

o Do we see the ACP objective in any DULL GRASP message from that interface? Diagnose the supported secure channel methods.

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

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

o If it failed, why:

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

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

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

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

    + The ACP domain certificate membership check (Section 6.1.3) fails:

      - The neighbor's certificate does not have the required trust anchor. Provide diagnostics which trust anchor 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?

o Are there one or more active ACP neighbors with secure channels?

o Is the RPL routing protocol for the ACP running?

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

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

o Is ACP GRASP running?

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

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

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

o ...

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

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

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

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

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

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

10.2. ACP Registrars

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

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

10.2.1. Registrar interactions

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

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

To orchestrate enrolling a candidate ACP node autonomically, the ACP registrar can rely on the ACP and use Proxies to reach the candidate ACP node, therefore allowing minimum pre-existing (auto-)configured network services on the candidate ACP node. BRSKI defines the BRSKI proxy, a design that can be adopted for various protocols that

Pledges/candidate ACP nodes could want to use, for example BRSKI over CoAP (Constrained Application Protocol), or proxying of Netconf.

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

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

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

An ACP registrar could operate EST for ACP certificate renewal and/or act as a CRL Distribution point. A node performing these services does not need to support performing (initial) enrollment, but it does require the same above described connectivity as an ACP registrar: via the ACP to ACP nodes and via the Data-Plane to the trust anchor and other sources of CRL information.

10.2.2. Registrar Parameter

The interactions of an ACP registrar outlined Section 6.10.7 and Section 10.2.1 above depend on the following parameters:

> A URL to the trust anchor (root CA) and credentials so that the ACP registrar can let the trust anchor sign candidate ACP member certificates.
>
> The ACP domain-name.

The Registrar-ID to use. This could default to a MAC address of the ACP registrar.

For recovery, the next-useable Node-IDs for zone (Zone-ID=0) sub-addressing scheme, for Vlong /112 and for Vlong /1120 sub-addressing scheme. These IDs would only need to be provisioned after recovering from a crash. Some other mechanism would be required to remember these IDs in a backup location or to recover them from the set of currently known ACP nodes.

Policies if candidate ACP nodes should receive a domain certificate or not, for example based on the devices IDevID as in BRSKI. The ACP registrar may have a whitelist or blacklist of devices "serialNumbers" from their IDevID.

Policies what type of address prefix to assign to a candidate ACP devices, based on likely the same information.

For BRSKI or other mechanisms using vouchers: Parameters to determine how to retrieve vouchers for specific type of secure bootstrap candidate ACP nodes (such as MASA URLs), unless this information is automatically learned such as from the IDevID of candidate ACP nodes (as defined in BRSKI).

10.2.3. Certificate renewal and limitations

When an ACP node renews/rekeys its certificate, it may end up doing so via a different registrar (e.g., EST server) than the one it originally received its ACP domain certificate from, for example because that original ACP registrar is gone. The ACP registrar through which the renewal/rekeying is performed would by default trust the ACP domain information from the ACP nodes current ACP domain certificate and maintain this information so that the ACP node maintains its ACP address prefix. In EST renewal/rekeying, the ACP nodes current ACP domain certificate is signaled during the TLS handshake.

This simple scenario has two limitations:

1. The ACP registrars cannot directly assign certificates to nodes and therefore needs an "online" connection to the trust anchor (root CA).

2. Recovery from a compromised ACP registrar is difficult. When an ACP registrar is compromised, it can insert for example conflicting ACP domain information and create thereby an attack against other ACP nodes through the ACP routing protocol.

Even when such a malicious ACP registrar is detected, resolving the problem may be difficult because it would require identifying all the wrong ACP domain certificates assigned via the ACP registrar after it was compromised. And without additional centralized tracking of assigned certificates there is no way to do this.

10.2.4. ACP Registrars with sub-CA

In situations, where either of the above two limitations are an issue, ACP registrars could also be sub-CAs. This removes the need for connectivity to a root-CA whenever an ACP node is enrolled, and reduces the need for connectivity of such an ACP registrar to a root-CA to only those times when it needs to renew its own certificate. The ACP registrar would also now use its own (sub-CA) certificate to enroll and sign the ACP nodes certificates, and therefore it is only necessary to revoke a compromised ACP registrars sub-CA certificate. Alternatively one can let it expire and not renew it, when the certificate of the sub-CA is appropriately short-lived.

As the ACP domain membership check verifies a peer ACP node's ACP domain certificate trust chain, it will also verify the signing certificate which is the compromised/revoked sub-CA certificate. Therefore ACP domain membership for an ACP node enrolled from a compromised and discovered ACP registrar will fail.

ACP nodes enrolled by a compromised ACP registrar would automatically fail to establish ACP channels and ACP domain certificate renewal via EST and therefore revert to their role as a candidate ACP members and attempt to get a new ACP domain certificate from an ACP registrar - for example, via BRSKI. In result, ACP registrars that have an associated sub-CA makes isolating and resolving issues with compromised registrars easier.

Note that ACP registrars with sub-CA functionality also can control the lifetime of ACP domain certificates easier and therefore also be used as a tool to introduce short lived certificates and not rely on CRL, whereas the certificates for the sub-CAs themselves could be longer lived and subject to CRL.

10.2.5. Centralized Policy Control

When using multiple, uncoordinated ACP registrars, several advanced operations are potentially more complex than with a single, resilient policy control backend, for example including but not limited to:

> Which candidate ACP node is permitted or not permitted into an ACP domain. This may not be a decision to be taken upfront, so that a per-"serialNumber" policy can be loaded into every ACP registrar.

Instead, it may better be decided in real-time including potentially a human decision in a NOC.

Tracking of all enrolled ACP nodes and their certificate information. For example in support of revoking individual ACP nodes certificates.

More flexible policies what type of address prefix or even what specific address prefix to assign to a candidate ACP node.

These and other operations could be introduced more easily by introducing a centralized Policy Management System (PMS) and modifying ACP registrar behavior so that it queries the PMS for any policy decision occurring during the candidate ACP node enrollment process and/or the ACP node certificate renewal process. For example, which ACP address prefix to assign. Likewise the ACP registrar would report any relevant state change information to the PMS as well, for example when a certificate was successfully enrolled onto a candidate ACP node.

10.3. Enabling and disabling ACP/ANI

Both ACP and BRSKI require interfaces to be operational enough to support sending/receiving their packets. In node types where interfaces are by default (e.g., without operator configuration) enabled, such as most L2 switches, this would be less of a change in behavior than in most L3 devices (e.g.: routers), where interfaces are by default disabled. In almost all network devices it is common though for configuration to change interfaces to a physically disabled state and that would break the ACP.

In this section, we discuss a suggested operational model to enable/ disable interfaces and nodes for ACP/ANI in a way that minimizes the risk of operator action to break the ACP in this way, and that also minimizes operator surprise when ACP/ANI becomes supported in node software.

10.3.1. Filtering for non-ACP/ANI packets

Whenever this document refers to enabling an interface for ACP (or BRSKI), it only requires to permit the interface to send/receive packets necessary to operate ACP (or BRSKI) - but not any other Data-Plane packets. Unless the Data-Plane is explicitly configured/ enabled, all packets not required for ACP/BRSKI should be filtered on input and output:

Both BRSKI and ACP require link-local only IPv6 operations on interfaces and DULL GRASP. IPv6 link-local operations means the minimum signaling to auto-assign an IPv6 link-local address and talk to neighbors via their link-local address: SLAAC (Stateless Address Auto-Configuration - [RFC4862]) and ND (Neighbor Discovery - [RFC4861]). When the device is a BRSKI pledge, it may also require TCP/TLS connections to BRSKI proxies on the interface. When the device has keying material, and the ACP is running, it requires DULL GRASP packets and packets necessary for the secure-channel mechanism it supports, e.g., IKEv2 and IPsec ESP packets or DTLS packets to the IPv6 link-local address of an ACP neighbor on the interface. It also requires TCP/TLS packets for its BRSKI proxy functionality, if it does support BRSKI.

10.3.2. Admin Down State

Interfaces on most network equipment have at least two states: "up" and "down". These may have product specific names. "down" for example could be called "shutdown" and "up" could be called "no shutdown". The "down" state disables all interface operations down to the physical level. The "up" state enables the interface enough for all possible L2/L3 services to operate on top of it and it may also auto-enable some subset of them. More commonly, the operations of various L2/L3 services is controlled via additional node-wide or interface level options, but they all become only active when the interface is not "down". Therefore an easy way to ensure that all L2/L3 operations on an interface are inactive is to put the interface into "down" state. The fact that this also physically shuts down the interface is in many cases just a side effect, but it may be important in other cases (see below, Section 10.3.2.2).

To provide ACP/ANI resilience against operators configuring interfaces to "down" state, this document recommends to separate the "down" state of interfaces into an "admin down" state where the physical layer is kept running and ACP/ANI can use the interface and a "physical down" state. Any existing "down" configurations would map to "admin down". In "admin down", any existing L2/L3 services of the Data-Plane should see no difference to "physical down" state. To ensure that no Data-Plane packets could be sent/received, packet filtering could be established automatically as described above in Section 10.3.1.

As necessary (see discussion below) new configuration options could be introduced to issue "physical down". The options should be provided with additional checks to minimize the risk of issuing them in a way that breaks the ACP without automatic restoration. For example they could be denied to be issued from a control connection (netconf/ssh) that goes across the interface itself ("do not disconnect yourself"). Or they could be performed only temporary and only be made permanent with additional later reconfirmation.

In the following sub-sections important aspects to the introduction of "admin down" state are discussed.

10.3.2.1. Security

Interfaces are physically brought down (or left in default down state) as a form of security. "Admin down" state as described above provides also a high level of security because it only permits ACP/ANI operations which are both well secured. Ultimately, it is subject to security review for the deployment whether "admin down" is a feasible replacement for "physical down".

The need to trust the security of ACP/ANI operations needs to be weighed against the operational benefits of permitting this: Consider the typical example of a CPE (customer premises equipment) with no on-site network expert. User ports are in physical down state unless explicitly configured not to be. In a misconfiguration situation, the uplink connection is incorrectly plugged into such as user port. The device is disconnected from the network and therefore no diagnostics from the network side is possible anymore. Alternatively, all ports default to "admin down". The ACP (but not the Data-Plane) would still automatically form. Diagnostics from the network side is possible and operator reaction could include to either make this port the operational uplink port or to instruct re-cabling. Security wise, only ACP/ANI could be attacked, all other functions are filtered on interfaces in "admin down" state.

10.3.2.2. Fast state propagation and Diagnostics

"Physical down" state propagates on many interface types (e.g., Ethernet) to the other side. This can trigger fast L2/L3 protocol reaction on the other side and "admin down" would not have the same (fast) result.

Bringing interfaces to "physical down" state is to the best of our knowledge always a result of operator action, but today, never the result of (autonomic) L2/L3 services running on the nodes. Therefore one option is to change the operator action to not rely on link-state propagation anymore. This may not be possible when both sides are under different operator control, but in that case it is unlikely that the ACP is running across the link and actually putting the interface into "physical down" state may still be a good option.

Ideally, fast physical state propagation is replaced by fast software driven state propagation. For example a DULL GRASP "admin-state" objective could be used to auto configure a Bidirectional Forwarding Protocol (BFD, [RFC5880]) session between the two sides of the link that would be used to propagate the "up" vs. admin down state.

Triggering physical down state may also be used as a mean of diagnosing cabling in the absence of easier methods. It is more complex than automated neighbor diagnostics because it requires coordinated remote access to both (likely) sides of a link to determine whether up/down toggling will cause the same reaction on the remote side.

See Section 10.1 for a discussion about how LLDP and/or diagnostics via GRASP could be used to provide neighbor diagnostics, and therefore hopefully eliminating the need for "physical down" for neighbor diagnostics - as long as both neighbors support ACP/ANI.

10.3.2.3. Low Level Link Diagnostics

"Physical down" is performed to diagnose low-level interface behavior when higher layer services (e.g., IPv6) are not working. Especially Ethernet links are subject to a wide variety of possible wrong configuration/cablings if they do not support automatic selection of variable parameters such as speed (10/100/1000 Mbps), crossover (Auto-MDIX) and connector (fiber, copper - when interfaces have multiple but can only enable one at a time). The need for low level link diagnostic can therefore be minimized by using fully auto configuring links.

In addition to "Physical down", low level diagnostics of Ethernet or other interfaces also involve the creation of other states on interfaces, such as physical Loopback (internal and/or external) or bringing down all packet transmissions for reflection/cable-length measurements. Any of these options would disrupt ACP as well.

In cases where such low-level diagnostics of an operational link is desired but where the link could be a single point of failure for the ACP, ASA on both nodes of the link could perform a negotiated diagnostics that automatically terminates in a predetermined manner without dependence on external input ensuring the link will become operational again.

10.3.2.4. Power Consumption Issues

Power consumption of "physical down" interfaces, may be significantly lower than those in "admin down" state, for example on long-range fiber interfaces. Bringing up interfaces, for example to probe reachability, may also consume additional power. This can make these type of interfaces inappropriate to operate purely for the ACP when they are not currently needed for the Data-Plane.

10.3.3. Interface level ACP/ANI enable

The interface level configuration option "ACP enable" enables ACP operations on an interface, starting with ACP neighbor discovery via DULL GRAP. The interface level configuration option "ANI enable" on nodes supporting BRSKI and ACP starts with BRSKI pledge operations when there is no domain certificate on the node. On ACP/BRSKI nodes, "ACP enable" may not need to be supported, but only "ANI enable". Unless overridden by global configuration options (see later), "ACP/ANI enable" will result in "down" state on an interface to behave as "admin down".

10.3.4. Which interfaces to auto-enable?

(Section 6.3) requires that "ACP enable" is automatically set on native interfaces, but not on non-native interfaces (reminder: a native interface is one that exists without operator configuration action such as physical interfaces in physical devices).

Ideally, ACP enable is set automatically on all interfaces that provide access to additional connectivity that allows to reach more nodes of the ACP domain. The best set of interfaces necessary to achieve this is not possible to determine automatically. Native interfaces are the best automatic approximation.

Consider an ACP domain of ACP nodes transitively connected via native interfaces. A Data-Plane tunnel between two of these nodes that are non-adjacent is created and "ACP enable" is set for that tunnel. ACP RPL sees this tunnel as just as a single hop. Routes in the ACP would use this hop as an attractive path element to connect regions adjacent to the tunnel nodes. In result, the actual hop-by-hop paths used by traffic in the ACP can become worse. In addition, correct forwarding in the ACP now depends on correct Data-Plane forwarding config including QoS, filtering and other security on the Data-Plane path across which this tunnel runs. This is the main issue why "ACP/ANI enable" should not be set automatically on non-native interfaces.

If the tunnel would connect two previously disjoint ACP regions, then it likely would be useful for the ACP. A Data-Plane tunnel could also run across nodes without ACP and provide additional connectivity for an already connected ACP network. The benefit of this additional ACP redundancy has to be weighed against the problems of relying on the Data-Plane. If a tunnel connects two separate ACP regions: how many tunnels should be created to connect these ACP regions reliably enough? Between which nodes? These are all standard tunneled network design questions not specific to the ACP, and there are no generic fully automated answers.

Instead of automatically setting "ACP enable" on these type of interfaces, the decision needs to be based on the use purpose of the non-native interface and "ACP enable" needs to be set in conjunction with the mechanism through which the non-native interface is created/ configured.

In addition to explicit setting of "ACP/ANI enable", non-native interfaces also need to support configuration of the ACP RPL cost of the link - to avoid the problems of attracting too much traffic to the link as described above.

Even native interfaces may not be able to automatically perform BRSKI or ACP because they may require additional operator input to become operational. Example include DSL interfaces requiring PPPoE credentials or mobile interfaces requiring credentials from a SIM card. Whatever mechanism is used to provide the necessary config to the device to enable the interface can also be expanded to decide on whether or not to set "ACP/ANI enable".

The goal of automatically setting "ACP/ANI enable" on interfaces (native or not) is to eliminate unnecessary "touches" to the node to make its operation as much as possible "zero-touch" with respect to ACP/ANI. If there are "unavoidable touches" such a creating/ configuring a non-native interface or provisioning credentials for a native interface, then "ACP/ANI enable" should be added as an option to that "touch". If a wrong "touch" is easily fixed (not creating another high-cost touch), then the default should be not to enable ANI/ACP, and if it is potentially expensive or slow to fix (e.g., parameters on SIM card shipped to remote location), then the default should be to enable ACP/ANI.

10.3.5. Node Level ACP/ANI enable

A node level command "ACP/ANI enable [up-if-only]" enables ACP or ANI on the node (ANI = ACP + BRSKI). Without this command set, any interface level "ACP/ANI enable" is ignored. Once set, ACP/ANI will operate 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".

10.3.5.1. Brownfield nodes

A "brownfield" node is one that already has a configured Data-Plane.

Executing global "ACP/ANI enable [up-if-only]" on each node is the only command necessary to create an ACP across a network of brownfield nodes once all the nodes have a domain certificate. When BRSKI is used ("ANI enable"), provisioning of the certificates only requires set-up of a single BRSKI registrar node which could also implement a CA for the network. This is the most simple way to introduce ACP/ANI into existing (== brownfield) networks.

The need to explicitly enable ACP/ANI is especially important in brownfield nodes because otherwise software updates may introduce support for ACP/ANI: Automatic enablement of ACP/ANI in networks where the operator does not only not want ACP/ANI but where the operator likely never even heard of it could be quite irritating to the operator. Especially when "down" behavior is changed to "admin down".

Automatically setting "ANI enable" on brownfield nodes where the operator is unaware of 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 network through the ACP that the attacker then has access to.

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.

10.3.5.2. Greenfield nodes

A "greenfield" node is one that did not have any prior configuration.

For greenfield nodes, only "ANI enable" is relevant. If another mechanism than BRSKI is used to (zero-touch) bootstrap a node, then it is up to that mechanism to provision domain certificates and to set global "ACP enable" as desired.

Nodes supporting full ANI functionality set "ANI enable" automatically when they decide that they are greenfield, e.g., that they are powering on from factory condition. They will then put all native interfaces into "admin down" state and start to perform BRSKI pledge functionality - and once a domain certificate is enrolled they automatically enable ACP.

Attempts for BRSKI pledge operations in greenfield state should terminate automatically when another method of configuring the node is used. Methods that indicate some form of physical possession of the device such as configuration via the serial console port could lead to immediate termination of BRSKI, while other parallel auto configuration methods subject to remote attacks might lead to BRSKI termination only after they were successful. Details of this may vary widely over different type of nodes. When BRSKI pledge operation terminates, this will automatically unset "ANI enable" and should terminate any temporarily needed state on the device to perform BRSKI - DULL GRASP, BRSKI pledge and any IPv6 configuration on interfaces.

10.3.6. Undoing ANI/ACP enable

Disabling ANI/ACP by undoing "ACP/ANI enable" is a risk for the reliable operations of the ACP if it can be executed by mistake or unauthorized. This behavior could be influenced through some additional (future) property in the certificate (e.g., in the domain information 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.

10.3.7. Summary

Node-wide "ACP/ANI enable [up-if-only]" commands enable the operation of ACP/ANI. This is only auto-enabled on ANI greenfield devices, otherwise it must be configured explicitly.

If the option "up-if-only" is not selected, interfaces enabled for ACP/ANI interpret "down" state as "admin down" and not "physical down". In "admin-down" all non-ACP/ANI packets are filtered, but the physical layer is kept running to permit ACP/ANI to operate.

(New) commands that result in physical interruption ("physical down", "loopback") of ACP/ANI enabled interfaces should be built to protect continuance or reestablishment of ACP as much as possible.

Interface level "ACP/ANI enable" control per-interface operations. It is enabled by default on native interfaces and has to be configured explicitly on other interfaces.

Disabling "ACP/ANI enable" global and per-interface should have additional checks to minimize undesired breakage of ACP. The degree of control could be a domain wide parameter in the domain certificates.

10.4. Configuration and the ACP (summary)

There is no desirable configuration for the ACP. Instead, all parameters that need to be configured in support of the ACP are limitations of the solution, but they are only needed in cases where not all components are made autonomic. Whereever 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 domain certificate, for example BRSKI, then such certificate needs to be configured via some pre-existing mechanisms outside the scope of this specification. Today, router have typically a variety of mechanisms to do this.

- Certificate maintenance requires PKI functions. Discovery of these functions across the ACP is automated (see Section 6.1.5), 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 10.3.

- When the ACP needs to be extended across interfacess other than L2, the ACP as defined in this document can not 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 aforemention non-automic components of the ACP, but in that case, the same caution needs to be exercised as with data-plane configuration without ACP: Misconfiguration may cause the configuring entity to be disconnected from the node it configures - for example when incorrectly unconfiguring a remote ACP neighbor through which the configured ACP node is reached.

11. Security Considerations

After seeding an ACP by configuring at least one ACP registrar with routing-subdomain and a CA, an ACP is self-protecting and there is no need to apply configuration to make it secure (typically the ACP Registrar doubles as EST server for certificate renewal). Its security therefore does not depend on configuration. This does not include workarounds for non-autonomic components as explained in Section 8. See Section 9.2 for details of how the ACP protects itself against attacks from the outside and to a more limited degree from the inside as well.

However, the security of the ACP depends on a number of other factors:

- o 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.

- o Every ACP registrar is criticial infrastructure that needs to be hardened against attacks, similar to a CA. A malicious registrar can enroll enemy plegdes to an ACP network or break ACP routing by duplicate ACP address assignment to pledges via their ACP domain certificates.

- Security can be compromised by implementation errors (bugs), as in all products.

There is no prevention of source-address spoofing inside the ACP. This implies that if an attacker gains access to the ACP, it can spoof all addresses inside the ACP and fake messages from any other node.

The ACP is designed to enable automation of current network management and future autonomic peer-to-peer/distributed network automation. Any ACP member can send ACP IPv6 packet to other ACP members and announce via ACP GRASP services to all ACP members without depenency against centralized components.

The ACP relies on peer-to-peer authentication and authorization using ACP certificates. This security model is necessary to enable the autonomic ad-hoc any-to-any connectivity between ACP nodes. It provides infrastructure protection through hop by hop authentication and encryption - without relying on third parties. For any services where this complete autonomic peer-to-peer group security model is appropriate, the ACP domain certificate can also be used unchanged. For example for any type of data-plane routing protocol security.

This ACP security model is designed primarily to protect against attack from the outside, but not against attacks from the inside. To protect against spoofing attacks from compromised on-path ACP nodes, end-to-end encryption inside the ACP is used by new ACP signaling: GRASP across the ACP using TLS. The same is expected from any non-legacy services/protocols using the ACP. Because no group-keys are used, there is no risk for impacted nodes to access end-to-end encrypted traffic from other ACP nodes.

Attacks from impacted ACP nodes against the ACP are more difficult than against the data-plane because of the autoconfiguration of the ACP and the absence of configuration options that could be abused that allow to change/break ACP behavior. This is excluding configuration for workaround in support of non-autonomic components.

Mitigation against compromised ACP members is possible through standard automated certificate management mechanisms including revocation and non-renewal of short-lived certificates. In this version of the specification, there are no further optimization of these mechanisms defined for the ACP (but see Appendix A.10.8).

Higher layer service built using ACP domain certificates should not solely rely on undifferentiated group security when another model is more appropriate/more secure. For example central network configuration relies on a security model where only few especially trusted nodes are allowed to configure the data-plane of network nodes (CLIL, Netconf). This can be done through ACP domain certificates by differentiating them and introduce roles. See Appendix A.10.5.

Fundamentally, security depends on avoiding operator and network operations automation mistakes, implementation and architecture. Autonomic approaches such as the ACP largely eliminate operator mistakes and make it easier to recover from network operations mistakes. Implementation and architectural mistakes are still possible, as in all networking technologies.

Many details of ACP are designed with security in mind and discussed elsewhere in the document:

IPv6 addresses used by nodes in the ACP are covered as part of the node's domain certificate as described in Section 6.1.2. 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 6.8.2.

ACP provides for secure, resilient zero-touch discovery of EST servers for certificate renewal. See Section 6.1.5.

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 6.5.

The ACP is designed to minimize attacks from the outside by minimizing its dependency against any non-ACP (Data-Plane) operations/configuration on a node. See also Section 6.12.2.

In combination with BRSKI, ACP enables a resilient, fully zero-touch network solution for short-lived certificates that can be renewed or re-enrolled even after unintentional expiry (e.g., because of interrupted connectivity). See Appendix A.2.

Because ACP secure channels can be long lived, but certificates used may be short lived, secure channels, for example built via IPsec need to be terminated when peer certificates expire. See Section 6.7.3.

The ACP is designed to minimize attacks from the outside by minimizing its dependency against any non-ACP (Data-Plane) operations/configuration on a node. See also Section 6.12.2.

12. IANA Considerations

This document defines the "Autonomic Control Plane".

The IANA is requested to register the value "AN_ACP" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.3.

The IANA is requested to register the value "SRV.est" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.1.5.

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 considered to be automatically registered in that registry for the same service for which <service-name> is registered according to [RFC6335]. This explanation is solely informational and has no impact on the requested registration.

The IANA is requested to create an ACP Parameter Registry with currently one registry table - the "ACP Address Type" table.

"ACP Address Type" Table. The value in this table are numeric values 0...3 paired with a name (string). Future values MUST be assigned using the Standards Action policy defined by [RFC8126]. The following initial values are assigned by this document:

0: ACP Zone Addressing Sub-Scheme (ACP RFC Figure 11) / ACP Manual Addressing Sub-Scheme (ACP RFC Section 6.10.4)
1: ACP Vlong Addressing Sub-Scheme (ACP RFC Section 6.10.5)

13. Acknowledgements

This work originated from an Autonomic Networking project at Cisco Systems, which started in early 2010. Many people contributed to this project and the idea of the Autonomic Control Plane, amongst which (in alphabetical order): Ignas Bagdonas, Parag Bhide, Balaji BL, Alex Clemm, Yves Hertoghs, Bruno Klauser, Max Pritikin, Michael Richardson, Ravi Kumar Vadapalli.

Special thanks to Brian Carpenter, Elwyn Davies, Joel Halpern and Sheng Jiang for their thorough reviews and to Pascal Thubert and Michael Richardson to provide the details for the recommendations of the use of RPL in the ACP.

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

14. Change log [RFC Editor: Please remove]

14.1. Initial version

First version of this document: draft-behringer-autonomic-control-plane

14.2. draft-behringer-anima-autonomic-control-plane-00

Initial version of the anima document; only minor edits.

14.3. draft-behringer-anima-autonomic-control-plane-01

- Clarified that the ACP should be based on, and support only IPv6.
- Clarified in intro that ACP is for both, between devices, as well as for access from a central entity, such as an NMS.
- Added a section on how to connect an NMS system.
- Clarified the hop-by-hop crypto nature of the ACP.
- Added several references to GDNP as a candidate protocol.
- Added a discussion on network split and merge. Although, this should probably go into the certificate management story longer term.

14.4. draft-behringer-anima-autonomic-control-plane-02

Addresses (numerous) comments from Brian Carpenter. See mailing list for details. The most important changes are:

- Introduced a new section "overview", to ease the understanding of the approach.

- o Merged the previous "problem statement" and "use case" sections into a mostly re-written "use cases" section, since they were overlapping.

- o Clarified the relationship with draft-ietf-anima-stable-connectivity

14.5. draft-behringer-anima-autonomic-control-plane-03

- o Took out requirement for IPv6 --> that's in the reference doc.

- o Added requirement section.

- o Changed focus: more focus on autonomic functions, not only virtual out-of-band. This goes a bit throughout the document, starting with a changed abstract and intro.

14.6. draft-ietf-anima-autonomic-control-plane-00

No changes; re-submitted as WG document.

14.7. draft-ietf-anima-autonomic-control-plane-01

- o Added some paragraphs in addressing section on "why IPv6 only", to reflect the discussion on the list.

- o Moved the Data-Plane ACP out of the main document, into an appendix. The focus is now the virtually separated ACP, since it has significant advantages, and isn't much harder to do.

- o Changed the self-creation algorithm: Part of the initial steps go into the reference document. This document now assumes an adjacency table, and domain certificate. How those get onto the device is outside scope for this document.

- o Created a new section 6 "workarounds for non-autonomic nodes", and put the previous controller section (5.9) into this new section. Now, section 5 is "autonomic only", and section 6 explains what to do with non-autonomic stuff. Much cleaner now.

- o Added an appendix explaining the choice of RPL as a routing protocol.

- o Formalized the creation process a bit more. Now, we create a "candidate peer list" from the adjacency table, and form the ACP with those candidates. Also it explains now better that policy (Intent) can influence the peer selection. (section 4 and 5)

- o Introduce a section for the capability negotiation protocol (section 7). This needs to be worked out in more detail. This will likely be based on GRASP.

- o Introduce a new parameter: ACP tunnel type. And defines it in the IANA considerations section. Suggest GRE protected with IPSec transport mode as the default tunnel type.

- o Updated links, lots of small edits.

14.8. draft-ietf-anima-autonomic-control-plane-02

- o Added explicitly text for the ACP channel negotiation.

- o Merged draft-behringer-anima-autonomic-addressing-02 into this document, as suggested by WG chairs.

14.9. draft-ietf-anima-autonomic-control-plane-03

- o Changed Neighbor discovery protocol from GRASP to mDNS. Bootstrap protocol team decided to go with mDNS to discover bootstrap proxy, and ACP should be consistent with this. Reasons to go with mDNS in bootstrap were a) Bootstrap should be reuseable also outside of full anima solutions and introduce as few as possible new elements. mDNS was considered well-known and very-likely even pre-existing in low-end devices (IoT). b) Using GRASP both for the insecure neighbor discovery and secure ACP operatations raises the risk of introducing security issues through implementation issues/non-isolation between those two instances of GRASP.

- o Shortened the section on GRASP instances, because with mDNS being used for discovery, there is no insecure GRASP session any longer, simplifying the GRASP considerations.

- o Added certificate requirements for ANIMA in section 5.1.1, specifically how the ANIMA information is encoded in subjectAltName.

- o Deleted the appendix on "ACP without separation", as originally planned, and the paragraph in the main text referring to it.

- o Deleted one sub-addressing scheme, focusing on a single scheme now.

- o Included information on how ANIMA information must be encoded in the domain certificate in section "preconditions".

- o Editorial changes, updated draft references, etc.

14.10. draft-ietf-anima-autonomic-control-plane-04

Changed discovery of ACP neighbor back from mDNS to GRASP after revisiting the L2 problem. Described problem in discovery section itself to justify. Added text to explain how ACP discovery relates to BRSKY (bootstrap) discovery and pointed to Michael Richardsons draft detailing it. Removed appendix section that contained the original explanations why GRASP would be useful (current text is meant to be better).

14.11. draft-ietf-anima-autonomic-control-plane-05

- Section 5.3 (candidate ACP neighbor selection): Add that Intent can override only AFTER an initial default ACP establishment.
- Section 6.10.1 (addressing): State that addresses in the ACP are permanent, and do not support temporary addresses as defined in RFC4941.
- Modified Section 6.3 to point to the GRASP objective defined in draft-carpenter-anima-ani-objectives. (and added that reference)
- Section 6.10.2: changed from MD5 for calculating the first 40-bits to SHA256; reason is MD5 should not be used any more.
- Added address sub-scheme to the IANA section.
- Made the routing section more prescriptive.
- Clarified in Section 8.1.1 the ACP Connect port, and defined that term "ACP Connect".
- Section 8.2: Added some thoughts (from mcr) on how traversing a L3 cloud could be automated.
- Added a CRL check in Section 6.7.
- Added a note on the possibility of source-address spoofing into the security considerations section.
- Other editoral changes, including those proposed by Michael Richardson on 30 Nov 2016 (see ANIMA list).

14.12. draft-ietf-anima-autonomic-control-plane-06

- Added proposed RPL profile.

o detailed DTLS profile - DTLS with any additional negotiation/ signaling channel.

o Fixed up text for ACP/GRE encap. Removed text claiming its incompatible with non-GRE IPsec and detailed it.

o Added text to suggest admin down interfaces should still run ACP.

14.13. draft-ietf-anima-autonomic-control-plane-07

o Changed author association.

o Improved ACP connect setion (after confusion about term came up in the stable connectivity draft review). Added picture, defined complete terminology.

o Moved ACP channel negotiation from normative section to appendix because it can in the timeline of this document not be fully specified to be implementable. Aka: work for future document. That work would also need to include analysing IKEv2 and describin the difference of a proposed GRASP/TLS solution to it.

o Removed IANA request to allocate registry for GRASP/TLS. This would come with future draft (see above).

o Gave the name "ACP domain information field" to the field in the certificate carrying the ACP address and domain name.

o Changed the rules for mutual authentication of certificates to rely on the domain in the ACP information field of the certificate instead of the OU in the certificate. Also renewed the text pointing out that the ACP information field in the certificate is meant to be in a form that it does not disturb other uses of the certificate. As long as the ACP expected to rely on a common OU across all certificates in a domain, this was not really true: Other uses of the certificates might require different OUs for different areas/type of devices. With the rules in this draft version, the ACP authentication does not rely on any other fields in the certificate.

o Added an extension field to the ACP information field so that in the future additional fields like a subdomain could be inserted. An example using such a subdomain field was added to the pre-existing text suggesting sub-domains. This approach is necessary so that there can be a single (main) domain in the ACP information field, because that is used for mutual authentication of the certificate. Also clarified that only the register(s) SHOULD/MUST use that the ACP address was generated from the domain name - so that we can easier extend change this in extensions.

- Took the text for the GRASP discovery of ACP neighbors from Brians grasp-ani-objectives draft. Alas, that draft was behind the latest GRASP draft, so i had to overhaul. The mayor change is to describe in the ACP draft the whole format of the M_FLOOD message (and not only the actual objective). This should make it a lot easier to read (without having to go back and forth to the GRASP RFC/draft). It was also necessary because the locator in the M_FLOOD messages has an important role and its not coded inside the objective. The specification of how to format the M_FLOOD message shuold now be complete, the text may be some duplicate with the DULL specificateion in GRASP, but no contradiction.

- One of the main outcomes of reworking the GRASP section was the notion that GRASP announces both the candidate peer's IPv6 link local address but also the support ACP security protocol including the port it is running on. In the past we shied away from using this information because it is not secured, but i think the additional attack vectors possible by using this information are negligible: If an attacker on an L2 subnet can fake another devices GRASP message then it can already provide a similar amount of attack by purely faking the link-local address.

- Removed the section on discovery and BRSKI. This can be revived in the BRSKI document, but it seems mood given how we did remove mDNS from the latest BRSKI document (aka: this section discussed discrepancies between GRASP and mDNS discovery which should not exist anymore with latest BRSKI.

- Tried to resolve the EDNOTE about CRL vs. OCSP by pointing out we do not specify which one is to be used but that the ACP should be used to reach the URL included in the certificate to get to the CRL storage or OCSP server.

- Changed ACP via IPsec to ACP via IKEv2 and restructured the sections to make IPsec native and IPsec via GRE subsections.

- No need for any assigned DTLS port if ACP is run across DTLS because it is signaled via GRASP.

14.14. draft-ietf-anima-autonomic-control-plane-08

Modified mentioning of BRSKI to make it consistent with current (07/2017) target for BRSKI: MASA and IDevID are mandatory. Devices with only insecure UDI would need a security reduced variant of BRSKI. Also added mentioning of Netconf Zero-Touch. Made BRSKI non-normative for ACP because wrt. ACP it is just one option how the domain certificate can be provisioned. Instead, BRSKI is mandatory when a device implements ANI which is ACP+BRSKI.

Enhanced text for ACP across tunnels to describe two options: one across configured tunnels (GRE, IPinIP etc) a more efficient one via directed DULL.

Moved decription of BRSKI to appendix to emphasize that BRSKI is not a (normative) dependency of GRASP, enhanced text to indicate other options how Domain Certificates can be provisioned.

Added terminology section.

Separated references into normative and non-normative.

Enhanced section about ACP via "tunnels". Defined an option to run ACP secure channel without an outer tunnel, discussed PMTU, benefits of tunneling, potential of using this with BRSKI, made ACP via GREP a SHOULD requirement.

Moved appendix sections up before IANA section because there where concerns about appendices to be too far on the bottom to be read. Added (Informative) / (Normative) to section titles to clarify which sections are informative and which are normative

Moved explanation of ACP with L2 from precondition to separate section before workarounds, made it instructive enough to explain how to implement ACP on L2 ports for L3/L2 switches and made this part of normative requirement (L2/L3 switches SHOULD support this).

Rewrote section "GRASP in the ACP" to define GRASP in ACP as mandatory (and why), and define the ACP as security and transport substrate to GRASP in ACP. And how it works.

Enhanced "self-protection" properties section: protect legacy management protocols. Security in ACP is for protection from outside and those legacy protocols. Otherwise need end-to-end encryption also inside ACP, e.g., with domain certificate.

Enhanced initial domain certificate section to include requirements for maintenance (renewal/revocation) of certificates. Added explanation to BRSKI informative section how to handle very short lived certificates (renewal via BRSKI with expired cert).

Modified the encoding of the ACP address to better fit RFC822 simple local-parts (":" as required by RFC5952 are not permitted in simple dot-atoms according to RFC5322. Removed reference to RFC5952 as its now not needed anymore.

Introduced a sub-domain field in the ACP information in the certificate to allow defining such subdomains with depending on future Intent definitions. It also makes it clear what the "main domain" is. Scheme is called "routing subdomain" to have a unique name.

Added V8 (now called Vlong) addressing sub-scheme according to suggestion from mcr in his mail from 30 Nov 2016 (https://mailarchive.ietf.org/arch/msg/anima/nZpEphrTqDCBdzsKMpaIn2gsIzI). Also modified the explanation of the single V bit in the first sub-scheme now renamed to Zone sub-scheme to distinguish it.

14.15. draft-ietf-anima-autonomic-control-plane-09

Added reference to RFC4191 and explained how it should be used on ACP edge routers to allow auto configuration of routing by NMS hosts. This came after review of stable connectivity draft where ACP connect is being referred to.

V8 addressing Sub-Scheme was modified to allow not only /8 device-local address space but also /16. This was in response to the possible need to have maybe as much as 2^12 local addresses for future encaps in BRSKI like IPinIP. It also would allow fully autonomic address assignment for ACP connect interfaces from this local address space (on an ACP edge device), subject to approval of the implied update to rfc4291/rfc4193 (IID length). Changed name to Vlong addressing sub-scheme.

Added text in response to Brian Carpenters review of draft-ietf-anima-stable-connectivity-04.

- o The stable connectivity draft was vaguely describing ACP connect behavior that is better standardized in this ACP draft.

- o Added new ACP "Manual" addressing sub-scheme with /64 subnets for use with ACP connect interfaces. Being covered by the ACP ULA prefix, these subnets do not require additional routing entries for NMS hosts. They also are fully 64-bit IID length compliant and therefore not subject to 4191bis considerations. And they avoid that operators manually assign prefixes from the ACP ULA prefixes that might later be assigned autonomically.

- o ACP connect auto-configuration: Defined that ACP edge devices, NMS hosts should use RFC4191 to automatically learn ACP prefixes.

This is especially necessary when the ACP uses multiple ULA prefixes (via e.g., the rsub domain certificate option), or if ACP connect sub-interfaces use manually configured prefixes NOT covered by the ACP ULA prefixes.

o Explained how rfc6724 is (only) sufficient when the NMS host has a separate ACP connect and Data-Plane interface. But not when there is a single interface.

o Added a separate subsection to talk about "software" instead of "NMS hosts" connecting to the ACP via the "ACP connect" method. The reason is to point out that the "ACP connect" method is not only a workaround (for NMS hosts), but an actual desirable long term architectural component to modularly build software (e.g., ASA or OAM for VNF) into ACP devices.

o Added a section to define how to run ACP connect across the same interface as the Data-Plane. This turns out to be quite challenging because we only want to rely on existing standards for the network stack in the NMS host/software and only define what features the ACP edge device needs.

o Added section about use of GRASP over ACP connect.

o Added text to indicate packet processing/filtering for security: filter incorrect packets arriving on ACP connect interfaces, diagnose on RPL root packets to incorrect destination address (not in ACP connect section, but because of it).

o Reaffirm security goal of ACP: Do not permit non-ACP routers into ACP routing domain.

Made this ACP document be an update to RFC4291 and RFC4193. At the core, some of the ACP addressing sub-schemes do effectively not use 64-bit IIDs as required by RFC4191 and debated in rfc4191bis. During 6man in Prague, it was suggested that all documents that do not do this should be classified as such updates. Add a rather long section that summarizes the relevant parts of ACP addressing and usage and. Aka: This section is meant to be the primary review section for readers interested in these changes (e.g., 6man WG.).

Added changes from Michael Richardsons review https://github.com/anima-wg/autonomic-control-plane/pull/3/commits, textual and:

o ACP discovery inside ACP is bad *doh*!.

o Better CA trust and revocation sentences.

o More details about RPL behavior in ACP.

o black hole route to avoid loops in RPL.

Added requirement to terminate ACP channels upon cert expiry/revocation.

Added fixes from 08-mcr-review-reply.txt (on github):

o AN Domain Names are FQDNs.

o Fixed bit length of schemes, numerical writing of bits (00b/01b).

o Lets use US american english.

14.16. draft-ietf-anima-autonomic-control-plane-10

Used the term routing subdomain more consistently where previously only subdomain was used. Clarified use of routing subdomain in creation of ULA "global ID" addressing prefix.

6.7.1.* Changed native IPsec encapsulation to tunnel mode (necessary), explained why. Added notion that ESP is used, added explanations why tunnel/transport mode in native vs. GRE cases.

6.10.3/6.10.5 Added term "ACP address range/set" to be able to better explain how the address in the ACP certificate is actually the base address (lowest address) of a range/set that is available to the device.

6.10.4 Added note that manual address sub-scheme addresses must not be used within domain certificates (only for explicit configuration).

6.12.5 Refined explanation of how ACP virtual interfaces work (p2p and multipoint). Did seek for pre-existing RFCs that explain how to build a multi-access interface on top of a full mesh of p2p connections (6man WG, anima WG mailing lists), but could not find any prior work that had a succinct explanation. So wrote up an explanation here. Added hopefully all necessary and sufficient details how to map ACP unicast packets to ACP secure channel, how to deal with ND packet details. Added verbiage for ACP not to assign the virtual interface link-local address from the underlying interface. Added note that GRAP link-local messages are treated specially but logically the same. Added paragraph about NBMA interfaces.

remaining changes from Brian Carpenters review. See Github file draft-ietf-anima-autonomic-control-plane/08-carpenter-review-reply.tx for more details:

Added multiple new RFC references for terms/technologies used.

Fixed verbage in several places.

2. (terminology) Added 802.1AR as reference.

2. Fixed up definition of ULA.

6.1.1 Changed definition of ACP information in cert into ABNF format. Added warning about maximum size of ACP address field due to domain-name limitations.

6.2 Mentioned API requirement between ACP and clients leveraging adjacency table.

6.3 Fixed TTL in GRASP example: msec, not hop-count!.

6.8.2 MAYOR: expanded security/transport substrate text:

Introduced term ACP GRASP virtual interface to explain how GRASP link-local multicast messages are encapsulated and replicated to neighbors. Explain how ACP knows when to use TLS vs. TCP (TCP only for link-local address (sockets). Introduced "ladder" picture to visualize stack.

6.8.2.1 Expanded discussion/explanation of security model. TLS for GRASP unicast connections across ACP is double encryption (plus underlying ACP secure channel), but highly necessary to avoid very simple man-in-the-middle attacks by compromised ACP members on-path. Ultimately, this is done to ensure that any apps using GRASP can get full end-to-end secrecy for information sent across GRASP. But for publically known ASA services, even this will not provide 100% security (this is discussed). Also why double encryption is the better/easier solution than trying to optimize this.

6.10.1 Added discussion about pseudo-random addressing, scanning-attacks (not an issue for ACP).

6.12.2 New performance requirements section added.

6.10.1 Added notion to first experiment with existing addressing schemes before defining new ones - we should be flexible enough.

6.3/7.2 clarified the interactions between MLD and DULL GRASP and specified what needs to be done (e.g., in 2 switches doing ACP per L2 port).

12. Added explanations and cross-references to various security aspects of ACP discussed elsewhere in the document.

13. Added IANA requirements.

Added RFC2119 boilerplate.

14.17. draft-ietf-anima-autonomic-control-plane-11

Same text as -10 Unfortunately when uploading -10 .xml/.txt to datatracker, a wrong version of .txt got uploaded, only the .xml was correct. This impacts the -10 html version on datatracker and the PDF versions as well. Because rfcdiff also compares the .txt version, this -11 version was created so that one can compare changes from -09 and changes to the next version (-12).

14.18. draft-ietf-anima-autonomic-control-plane-12

Sheng Jiangs extensive review. Thanks! See Github file draft-ietf-anima-autonomic-control-plane/09-sheng-review-reply.txt for more details. Many of the larger changes listed below where inspired by the review.

Removed the claim that the document is updating RFC4291,RFC4193 and the section detailing it. Done on suggestion of Michael Richardson - just try to describe use of addressing in a way that would not suggest a need claim update to architecture.

Terminology cleanup:

- o Replaced "device" with "node" in text. Kept "device" only when referring to "physical node". Added definitions for those words. Includes changes of derived terms, especially in addressing: "Node-ID" and "Node-Number" in the addressing details.

- o Replaced term "autonomic FOOBAR" with "acp FOOBAR" as wherever appropriate: "autonomic" would imply that the node would need to support more than the ACP, but that is not correct in most of the cases. Wanted to make sure that implementers know they only need to support/implement ACP - unless stated otherwise. Includes "AN->ACP node", "AN->ACP adjacency table" and so on.

1 Added explanation in the introduction about relationship between ACP, BRSKI, ANI and Autonomic Networks.

6.1.1 Improved terminology and features of the certificate information field. Now called domain information field instead of ACP information field. The acp-address field in the domain information field is now optional, enabling easier introduction of various future options.

6.1.2 Moved ACP domain membership check from section 6.6 to (ACP secure channels setup) here because it is not only used for ACP secure channel setup.

6.1.3 Fix text about certificate renewal after discussion with Max Pritikin/Michael Richardson/Brian Carpenter:

- o Version 10 erroneously assumed that the certificate itself could store a URL for renewal, but that is only possible for CRL URLs. Text now only refers to "remembered EST server" without implying that this is stored in the certificate.

- o Objective for RFC7030/EST domain certificate renewal was changed to "SRV.est" See also IANA section for explanation.

- o Removed detail of distance based service selection. This can be better done in future work because it would require a lot more detail for a good DNS-SD compatible approach.

- o Removed detail about trying to create more security by using ACP address from certificate of peer. After rethinking, this does not seem to buy additional security.

6.10 Added reference to 6.12.5 in initial use of "loopback interface" in section 6.10 in result of email discussion michaelR/michaelB.

10.2 Introduced informational section (diagnostics) because of operational experience - ACP/ANI undeployable without at least diagnostics like this.

10.3 Introduced informational section (enabling/disabling) ACP. Important to discuss this for security reasons (e.g., why to never auto-enable ANI on brownfield devices), for implementers and to answer ongoing questions during WG meetings about how to deal with shutdown interface.

10.8 Added informational section discussing possible future variations of the ACP for potential adopters that cannot directly use the complete solution described in this document unmodified.

14.19. draft-ietf-anima-autonomic-control-plane-13

Swap author list (with permission).

6.1.1. Eliminate blank lines in definition by making it a picture (reformatting only).

6.10.3.1 New paragraph: Explained how nodes using Zone-ID != 0 need to use Zone-ID != 0 in GRASP so that we can avoid routing/forwarding of Zone-ID = 0 prefixes.

Rest of feedback from review of -12, see https://raw.githubusercontent.com/anima-wg/autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/12-feedback-reply.txt

Review from Brian Carpenter:

various: Autonomous -> autonomic(ally) in all remaining occurrences.

various: changed "manual (configured)" to "explicitly (configured)" to not exclude the option of (SDN controller) automatic configuration (no humans involved).

1. Fixed reference to section 9.

2. Added definition of loopback interface == internal interface. After discus on WG mailing lists, including 6man.

6.1.2 Defined CDP/OCSP and pointed to RFC5280 for them.

6.1.3 Removed "EST-TLS", no objective value needed or beneficial, added explanation paragraph why.

6.2 Added to adjacency table the interface that a neighbor is discovered on.

6.3 Simplified CDDL syntax: Only one method per AN_ACP objective (because of locators). Example with two objectives in GRASP message.

6.8.1 Added note about link-local GRASP multicast message to avoid confusion.

8.1.4 Added RFC8028 as recommended on hosts to better support VRF-select with ACP.

8.2.1 Rewrote and Simplified CDDL for configured remote peer and explanations. Removed pattern option for remote peer. Not important enough to be mandated.

Review thread started by William Atwood:

2. Refined definition of VRF (vs. MPLS/VPN, LISP, VRF-LITE).

2. Refined definition of ACP (ACP includes ACP GRASP instance).

2. Added explanation for "zones" to terminology section and into Zone Addressing Sub Scheme section, relating it to RFC4007 zones (from Brian Carpenter).

4. Fixed text for ACP4 requirement (Clients of the ACP must not be tied to specific protocol.).

5. Fixed step 4. with proposed text.

6.1.1 Included suggested explanation for rsub semantics.

6.1.3 must->MUST for at least one EST server in ACP network to autonomically renew certs.

6.7.2 normative: AND MUST NOT (permit weaker crypto options.

6.7.1.1 also included text denying weaker IPsec profile options.

6.8.2 Fixed description how to build ACP GRASP virtual interfaces. Added text that ACP continues to exist in absence of ACP neighbors.

various: Make sure all "zone" words are used consistently.

6.10.2/various: fixed 40-bit RFC4193 ULA prefix in all examples to 89b714f3db (thanks MichaelR).

6.10.1 Removed comment about assigned ULA addressing. Decision not to use it now ancient history of WG decision making process, not worth nothing anymore in the RFC.

Review from Yongkang Zhang:

6.10.5 Fixed length of Node-Numbers in ACP Vlong Addressing Sub-Scheme.

14.20. draft-ietf-anima-autonomic-control-plane-14

Disclaimer: All new text introduced by this revision provides only additional explanations/ details based on received reviews and analysis by the authors. No changes to behavior already specified in prior revisions.

Joel Halpern, review part 3:

Define/explain "ACP registrar" in reply to Joel Halpern review part 3, resolving primarily 2 documentation issues::

1. Unclear how much ACP depends on BRSKI. ACP document was referring unqualified to registrars and Registrar-ID in the addressing section without explaining what a registrar is, leading to the assumption it must be a BRSKI Registrar.

2. Unclear how the ACP addresses in ACP domain certificates are assigned because the BRSKI document does not defines this, but refers to this ACP document.

Wrt. 1: ACP does NOT depend on BRSKI registrars, instead ANY appropriate automated or manual mechanism can be used to enroll ACP nodes with ACP domain certificates. This revision calls defines such mechanisms the "ACP registrar" and defines requirements. this is non-normative, because it does not define specific mechanisms that need to be support. In ANI devices, ACP Registrars are BRSKI Registrars. In non-ANI ACP networks, the registrar may simply be a person using CLI/web-interfaces to provision domain certificates and set the ACP address correctly in the ACP domain certificate.

Wrt. 2.: The BRSKI document does rightfully not define how the ACP address assignment and creation of the ACP domain information field has to work because this is independent of BRSKI and needs to follow the same rules whatever protocol/mechanisms are used to implement an ACP Registrar. Another set of protocols that could be used instead of BRSKI is Netconf/Netconf-Call-Home, but such an alternative ACP Registrar solution would need to be specified in its own document.

Additional text/sections had to be added to detail important conditions so that automatic certificate maintenance for ACP nodes (with BRSKI or other mechanisms) can be done in a way that as good as possible maintains ACP address information of ACP nodes across the nodes lifetime because that ACP address is intended as an identifier of the ACP node.

Summary of sections added:

- o 6.1.3.5/6.1.3.6 (normative): re-enrollment of ACP nodes after certificate expiry/failure in a way that allows to maintain as much as possible ACP address information.

- o 6.10.7 (normative): defines "ACP Registrar" including requirements and how it can perform ACP address assignment.

- o 10.3 (informative): details / examples about registrars to help implementers and operators understand easier how they operate, and provide suggestion of models that a likely very useful (sub-CA and/or centralized policy management).

- o 10.4 (informative): Explains the need for the multiple address sub-spaces defined in response to discuss with Joel.

Other changes:

Updated references (RFC8366, RFC8368).

Introduced sub-section headings for 6.1.3 (certificate maintenance) because section became too long with newly added sub-sections. Also some small text fixups/remove of duplicate text.

Gen-ART review, Elwyn Davies:

[RFC Editor: how can i raise the issue of problematic cross references of terms in the terminology section - rendering is problematic.].

4. added explanation for ACP4 (finally).

6.1.1 Simplified text in bullet list explaining rfc822 encoding.

6.1.3 refined second paragraph defining remembering of previous EST server and explaining how to do this with BRSKI.

9.1 Added paragraph outlining the benefit of the sub-CA Registrar option for supporting partitioned networks.

Roughly 100 more nits/minor fixes throughout the document. See: https://raw.githubusercontent.com/anima-wg/autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/13-elwynd-reply.txt

Joel Halpern, review part 2:

6.1.1: added note about "+ +" format in address field when acp-address and rsub are empty.

6.5.10 - clarified text about V bit in Vlong addressing scheme.

6.10.3/6.10.4 - moved the Z bit field up front (directly after base scheme) and indicated more explicitly Z is part of selecting of the sub-addressing scheme.

Refined text about reaching CRL Distribution Point, explain why address as indicator to use ACP.

Note from Brian Carpenter: RFC Editor note for section reference into GRASP.

IOT directorate review from Pascal Thubert:

Various Nits/typos.

TBD: Punted wish for mentioning RFC reference titles to RFC editor for now.

1. Added section 1.1 - applicability, discussing protocol choices re. applicability to constrained devices (or not). Added notion of TCP/TLS via CoAP/DTLS to section 10.4 in support of this.

2. Added in-band / out-of-band into terminology.

5. Referenced section 8.2 for remote ACP channel configuration.

6.3 made M_FLOOD periods RECOMMENDED (less guesswork)

6.7.x Clarified conditional nature of MUST for the profile details of IPsec parameters (aka: only 6.7.3 defines actual MUST for nodes, prior notions only define the requirements for IPsec profiles IF IPsec is supported.

6.8.1 Moved discussion about IP multicast, IGP, RPL for GRASP into a new subsection in the informative part (section 10) to tighten up text in normative part.

6.10.1 added another reference to stable-connectivity for interop with IPv4 management.

6.10.1 removed mentioning of ULA-Random, term was used in email discus of ULA with L=1, but term actually not defined in rfc4193, so mentioning it is just confusing/redundant. Also added note about the random hash being defined in this document, not using SHA1 from rfc4193.

6.11.1.1 added suggested text about mechanisms to further reduce opportunities for loop during reconvergence (active signaling options from RFC6550).

6.11.1.3 made mode 2 MUST and mode 2 MAY (RPL MOP - mode of operations). Removes ambiguity.

6.12.5 Added recommendation for RFC4429 (optimistic DAD).

Nits from Benjamin Kaduk: dTLS -> DTLS:

Review from Joel Halpern:

1. swapped order of "purposes" for ACP to match order in section 3.

1. Added notion about manageability of ACP gong beyond RFC7575 (before discussion of stable connectivity).

2. Changed definition of Intent to be same as reference model (policy language instead of API).

6.1.1 changed BNF specification so that a local-part without acp-address (for future extensions) would not be rfcSELF.+rsub but simpler rfcSELF+rsub. Added explanation why rsub is in local-part.

Tried to eliminate unnecessary references to VRF to minimize assumption how system is designed.

6.1.3 Explained how to make CDP reachable via ACP.

6.7.2 Made it clearer that constrained devices MUST support DTLS if they cannot support IPsec.

6.8.2.1 clarified first paragraph (TCP retransmissions lightweight).

6.11.1 fixed up RPL profile text - to remove "VRF". Text was also buggy. mentioned control plane, but it’s a forwarding/silicon issue to have these header.

6.12.5 Clarified how link-local ACP channel address can be derived, and how not.

8.2.1 Fixed up text to distinguish between configuration and model describing parameters of the configuration (spec only provides parameter model).

Various Nits.

14.21. draft-ietf-anima-autonomic-control-plane-15

Only reshuffling and formatting changes, but wanted to allow reviewers later to easily compare -13 with -14, and these changes in -15 mess that up too much.

increased TOC depth to 4.

Separated and reordered section 10 into an operational and a background and futures section. The background and futures could also become appendices if the layout of appendices in RFC format wasn't so horrible that you really only want to avoid using them (all the way after a lot of text like references that stop most readers from proceeding any further).

14.22. draft-ietf-anima-autonomic-control-plane-16

Mirja Kuehlewind:

Tightened requirements for ACP related GRASP objective timers.

Better text to introduce/explains baseline and constrained ACP profiles.

IANA guideline: MUST only accept extensible last allocation for address sub-scheme.

Moved section 11 into appendix.

Warren Kumari:

Removed "global routing table", replaced with "Data-Plane routing (and forwarding) tables.

added text to indicate how routing protocols do like to have data-plane dependencies.

Changed power consumption section re. admin-down state. Power needed to bring up such interfaces make t inappropriate to probe. Need to think more about best suggests -> beyond scope.

Replaced "console" with out-of-band... (console/management ethernet).

Various nits.

Joel Halpern:

Fixed up domain information field ABNF to eliminate confusion that rsub is not an FQDN but only a prefix to routing-subdomain.

Corrected certcheck to separate out cert verification into lifetime validity and proof of ownership of private key.

Fixed pagination for "ACP as security and transport substrate for GRASP" picture.

14.23. draft-ietf-anima-autonomic-control-plane-17

Review Alissa Cooper:

Main discuss point fixed by untangling two specific node type cases:

NOC nodes have ACP domain cert without acp-address field. Are ACP domain members, but cannot build ACP secure channels (just end-to-end or nay other authentications.

ACP nodes may have other methods to assign ACP address than getting it through the cert. This is indicated through new value 0 for acp-address in certificate.

Accordingly modified texts in ABNF/explanation and Cert-Check section.

Other:

Better separation of normative text and considerations for "future" work:

- Marked missing chapters as Informative. Reworded requirements section to indicate its informative nature, changed requirements to _MUST_/_SHOULD_ to indicate these are not RFC2119 requirements but that this requirements section is really just in place of a separate solutions requirements document (that ANIMA was not allowed to produce).

- removed ca. 20 instances of "futures" in normative part of document.

- moved important instances of "futures" into new section A.10 (last section of appendix). These serve as reminder of work discussed during WG but not able to finish specifying it.

Eliminated perception that "rsub" (routing subdomain) is only beneficial with future work. Example in A.7.

Added RFC-editor note re formatting of references to terms defined in terminology section.

Using now correct RFC 8174 boilerplate.

Clarified semantic and use of manual ACP sub-scheme. Not used in certificates, only assigned via traditional methods. Use for ACP-connect subnets or the like.

Corrected text about Data-Plane dependencies of ACP. Appropriate implementations can be fully data-plane independent (without more spec work) if not sharing link-local address with Data-Plane. 6.12.2 text updated to discuss those (MAC address), A.10.2 discusses options that would require new standards work.

Moved all text about Intent into A.8 to clearly mark it as futures.

Changed suggestion of future insecure ACP option to future "end-to-end-security-only" option.

Various textual fixes.

Gen-ART review by Elwyn Davies:

Some fixes also mentioned by Alissa.

Added reference for OT.

Fixed notion that secure channel is not only a security association.

>20 good textual fixes. Thanks!

Other:

Added picture requested by Pascal Thubert about Dual-NOC (A.10.4).

Moved RFC-editor request for better first RFC reference closer to the top of the document.

Fixed typo /126 -> 127 for prefix length with zone address scheme.

Overlooked early SecDir review from frank.xialiang@huawei.com:

most issues fixed through other review in -16. Added reference to self-protection section 9.2 into security considerations section.

14.24. draft-ietf-anima-autonomic-control-plane-18

Too many word/grammar mistakes in -17.

14.25. draft-ietf-anima-autonomic-control-plane-19

Review Eric Rescola:

6.1.2 - clarified that we do certificate path validation against potentially multiple trust anchors.

6.1.3 - Added more comprehensive explanation of Trust Points via new section 6.1.3.

6.5 - added figure with sequential steps of ACP channel establishment and Alice and Bob finding their role in the setup.

6.7.x - detailled crypto profiles: AES-256-GCM, ECDHE.

6.7.2 - Referring to RFC7525 as the required crypto profile for DTLS (taking text from RFC8310 as previously discussed with Eric).

6.7.3 - Added explanation that ACP needs no single MTI secure channel protocol with example.

6.10.2 - Added requirement that rsub must be choosen so that they don't create SHA256 collisions. Added explanation how the same could be done for different ACP networks with same trust anchors but that this outside the scope of this specification.

6.7.10 - Explains security expectations against ACP registrars: Must be trusted and then given credentials to act as PKI RA to help pledges to enroll with an ACP certificate.

9.1 - Added explanations about merging ACP domains requiring both domains to trust union of Trust Anchors and need to avod ULA hash collisions.

11 - Added that ACP registrars are critical infrastructure requiring hardening like CA, mentioning attack impact examples.

11 - Mentioning that ACP requires initial setup of CA and registrar.

11 - long rewrite/extension of group security model and its implication shared with review from Ben (below).

Many nits fixed.

Review Benjamin Kaduk:

Fixed various nits.

Changed style of MUST/SHOULD in Requirements section to all lower case to avoid any RFC2119 confusion.

1. clarified support for constrained devices/DTLS: Opportunistic.

1. Clarified ACPs use of two variants of GRASP DULL for neighbor discovery and ACP grasp for service discovery/clients.

3.2 - amended text explaining what additional security ACP provides for bootstrap protocols.

6.1.1 - Added note about ASN.1 encoding in the justification for use of rfc822address.

6.1.2 - Added details how to handle ACP connection when node via which OCSP/CRL-server is reached fails certificate verification.

12. Rewrote explanation why objective names requested for ACP use SRV.name.

10.4 - added summary section about ACP and configuration.

Review Eric Rescorla:

6.1.2 - changed peer certificate verification to be certificate path verification, added lowercase normalizaion comparison to domain name check.

6.1.2 - explained how domain membership check is authentication and authorization.

6.1.4.1 - Fixed "objective value" to "objective name".

6.1.4.3 - check IPv6 address of CDP against CDP ACP certificate IPv6 address only if URL uses IPv6 address.

6.10.1 - added more justification why there is no need for privacy protection of ACP addresses.

6.11.1.1 - thorough fixup of sentences/structure of this RPL overview section to make it more logical and easier to digest. Also added a paragraph about the second key benefit of this profile (scalability).

6.11.1.9 - Added explanation about not using RPL security from Benjamin.

8.1.1 - Fixed up text for address assignment of ACP connect interfaces. Only recommending manual addressing scheme.

9.1 - changed self-healing benefit text to describe immediate channel reset for short-lived certificates and describing how the same with CRL/OCSP is optional.

11. - added note about immediate termination of secure channels after certificate expiry as this is uncommon today.

11. - rewrote section of security model, attacks and mitigation of compromised ACP members.

A.24 - clarified the process in which expired certificates are used for certificate renewal to avvoid higher overhead of -re-enrolment.

A.4 - removed mentioning of RPL trickle because not used by ACP RPL profile.

A.10.8 - added section discussing how to minimize risk of compromised nodes, recovering them or kicking them out.

14.26. Open Issues in -19

Need to find good reference for TLS profile for ACP GRASP TLS connections.

TBD: Add DTLS choice to GRASP secure channel.

14.27. draft-ietf-anima-autonomic-control-plane-20

(1)In reply to review of -16 by Ben Kaduk:

Diff:

http://tools.ietf.org/tools/rfcdiff/
rfcdiff.pyht?url1=https://raw.githubusercontent.com/anima-wg/
autonomic-control-plane/master/draft-ietf-anima-autonomic-control-
plane/draft-ietf-anima-autonomic-control-plane-
19.txt&url2=https://raw.githubusercontent.com/anima-wg/autonomic-
control-plane/master/draft-ietf-anima-autonomic-control-plane/draft-
ietf-anima-autonomic-control-plane.19.1.txt

6.1.1 - Changed ABNF to use HEXDIG (to simpllify ABNF) and changed example to resulting uppercase hex characters. There was no specific reason to pick lower vs. upper-case left, and predefined HEXDIG is uppercase.

- Added explanation why internationalized domain-names are not supported/required.

6.1.4.1 - better hint/formal text to explain grasp objective encoding parameters.

6.7.1.1 - rewrote IPsec/IKEv2 requirements after discovering the appropriate references RFC8221/8247 and ben asking to use IANA registry code-point words correctly. Fully relying on RFC recommendatin parameters, just stripping down required options to minimum for IPsec to simplify any HW dependencies. Not stripped down IKEv2 (SW) requirements though - unclear if that would be useful.

6.7.1.2 - stripped down IPsec/GRE requirements by referring to the new text in 6.7.1.1 (reducing duplication that existed up to -19.

Referring to RFC7525 also as BCP as requested by Benj. Q: Do BCPs keep numbers even if RFCs for them change ?? Otherwise i am not sure what difference it makes to mention the BCP (maybe just to emphasize on the operartional character..).

6.8.2 - Also referring now to RFC7525 for TLS requirements. Authors had overlooked that it was not only covering DTLS, but also TLS.

6.11.1.14 - added paragraph about "simple" nodes not having to become RPL roots (and therefore have less forwarding plane requirements. Logic is that when there are only "simple" nodes, the requested forwarding plane feature (diagnostcs) wouldn't be useful immediately anyhow because no operator could connect to the network (all access to ACP assumed to be via more intelligent nodes.

10.3.7 - Refined/amended explanation of ACP-connect configuration case and how it can also be simple.

various - fixed places where LDevID was used instead of IDevID. Some other smaller textual fixes

(2)In reply to review of -16 by Eric Rescorla (by Ben Kaduk):

Diff:

http://tools.ietf.org/tools/rfcdiff/
rfcdiff.pyht?url1=https://raw.githubusercontent.com/anima-wg/
autonomic-control-plane/master/draft-ietf-anima-autonomic-control-
plane/draft-ietf-anima-autonomic-control-

plane.19.1.txt&url2=https://raw.githubusercontent.com/anima-wg/autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/draft-ietf-anima-autonomic-control-plane.19.2.txt

6.1.1. - new subsection to cover generic ACP certificate requirements. Must be rfc5280 complaint. Asks for what is understood to be hopefully best current practices: Certificate with ECDH public key, preferrably signed by ECDH key, describes how ACP domain information field is effectively non cryptographic identity/ name of entity owning certificate.

6.1.3 (was 6.1.2) - ACP domain membership text refinement:

emphasises how this includes action of a security association protocol (example IKEv2 proof of ownership of cert), reines relevant parts of rfc5280, refines that CRL/OCSP are skipped only in absence of doing them via secure association protocol)..

created new summary at the end restating the purpose of the different steps.

6.1.5 (was 6.1.4) - added requirement for EST server certificate to have extended key use id-kp-cmcRA so clients can trust them when requesting refreshed CA certificates. Also explains how to set up EST server for multiple ACP domains.

6.3 - DULL GRASP, explains why certs are not signalled in DULL grasp.

6.7 - explains generic requirements against security association protocols: be able to authenticate with ACP certificates, have appropriate security level (weakest link). Be able to directly signal ACP certificates due to absence of other option to learn peer cert. Discusses cases (L2 security, physical security) where security association protocol security can be relaxed/removed).

8.1.5 - GRASP via ACP connect: removed suggestion for policy filtering of GRASP messages. Better restatement of security model of ACP connect to argue that GRASP is to be run equally run across it as across the rest of ACP.

few smaller textual improvements.

In reply to 2nd review email (new ballot position) by Ben Kaduk:

Diff:

http://tools.ietf.org/tools/rfcdiff/rfcdiff.pyht?url1=https://raw.githubusercontent.com/anima-wg/

autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/draft-ietf-anima-autonomic-control-plane.19.2.txt&url2=https://raw.githubusercontent.com/anima-wg/autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/draft-ietf-anima-autonomic-control-plane.19.3.txt

large number of textual nits (thanks a lot!).

6.1.1 - Attempted to further complete whats required in certificate: take from rfc5280 and if feasible by operator policy copy attributes from IDevID for easier diagnostics. Added note that authorization mechanisms for ACP can extend over time leveraging other fields of certificate.

6.1.2 - Added paragraph stating ABNF domain-information field is what becomes the rfc822address and other ABNF terminal nodes are used just in the text (and hash creation).

6.1.5.1 - added note about unknown GRASP objective values MUST be ignored.

6.1.5.3 - CDP distribution uses HTTP, not HTTP (to avoid circular authentication reuirements and because payload is self signed/authenticated).

6.5 (Channel selection) - Alice/Bob uses ACP address as tiebreakers, empty ACP address just means "lowest tie-breaker" (become Bob).

10.3.5.1 - Better justification why ANI MUST be explicitly configured on brownfield nodes - customer not bothering about registering company with MASA etc.

A.6 - added RFC editor note to remove this section before publication (as i think we agreed on earlier in WG).

A.7 - added note about possible security wise undesirability to make IDevID information of nodes available even easier through unprotected protocols.

In reply to Michael Richardson:

Diff:

http://tools.ietf.org/tools/rfcdiff/rfcdiff.pyht?url1=https://raw.githubusercontent.com/anima-wg/autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/draft-ietf-anima-autonomic-control-plane.19.3.txt&url2=https://raw.githubusercontent.com/anima-wg/

autonomic-control-plane/master/draft-ietf-anima-autonomic-control-plane/draft-ietf-anima-autonomic-control-plane.19.4.txt

Missing changes lost in github from March 2019, no functional all textual/representation improvements:

Terminology text improvements, sUDI, routing-subdomain.

6.10.2 - new table showing the different address formats as a summary.

6.10.5 - Vlong address format, cleaned up representation by introducing F-bit to distinguish 23/15 Vlong address prefixes. Adjusted explanatory text accordingly.

6.1.5.1 - Changed TCP port for SRV.est from 80 to 443 (EST is using TLS).

Other:

Change author association: Toerless Eckert, Huawei -> Futurewei USA.

14.28. draft-ietf-anima-autonomic-control-plane-21

In reply to review of -19 by Ben Kaduk (also for Eric Rescorla) that where not fixed in -20:

6.1.1 Added missing detail about ACP domain certificate requirements and explanations.

Also added pointer to CABFORUM cert recommendations as suggested by MichaelR (changing pointer, external non-IETF document, so can not be a real MUST/SHOULD).

6.7 Added requirement for PFS in secure channel protocols.

6.8.2 Added TLS security profile based on 'intersection' of Bens recommended profiles and RFC7525.

15. References

15.1. Normative References

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

[I-D.ietf-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR and JSON data structures", draft-ietf-cbor-cddl-08 (work in progress), March 2019.

[IKEV2IANA]
IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <https://www.rfc-editor.org/info/rfc4106>.

[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>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <https://www.rfc-editor.org/info/rfc5322>.

[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>.

[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>.

15.2. Informative References

[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", Nov 2019, <https://cabforum.org/baseline-requirements-certificate-contents/>.

[I-D.eckert-anima-noc-autoconfig]
Eckert, T., "Autoconfiguration of NOC services in ACP networks via GRASP", draft-eckert-anima-noc-autoconfig-00 (work in progress), July 2018.

[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)", draft-ietf-acme-star-11 (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-29 (work in progress), October 2019.

[I-D.ietf-anima-prefix-management]
Jiang, S., Du, Z., Carpenter, B., and Q. Sun, "Autonomic IPv6 Edge Prefix Management in Large-scale Networks", draft-ietf-anima-prefix-management-07 (work in progress), December 2017.

[I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-10 (work in progress), November 2018.

[I-D.ietf-netconf-zerotouch]
Watsen, K., Abrahamsson, M., and I. Farrer, "Secure Zero Touch Provisioning (SZTP)", draft-ietf-netconf-zerotouch-29 (work in progress), January 2019.

[I-D.ietf-roll-applicability-template]
Richardson, M., "ROLL Applicability Statement Template", draft-ietf-roll-applicability-template-09 (work in progress), May 2016.

[I-D.ietf-roll-useofrplinfo]
Robles, I., Richardson, M., and P. Thubert, "Using RPL Option Type, Routing Header for Source Routes and IPv6-in-IPv6 encapsulation in the RPL Data Plane", draft-ietf-roll-useofrplinfo-31 (work in progress), August 2019.

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-33 (work in progress), October 2019.

[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., "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>.

[RFC1886] Thomson, S. and C. Huitema, "DNS Extensions to support IP version 6", RFC 1886, DOI 10.17487/RFC1886, December 1995, <https://www.rfc-editor.org/info/rfc1886>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., 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>.

[RFC2821] Klensin, J., Ed., "Simple Mail Transfer Protocol", RFC 2821, DOI 10.17487/RFC2821, April 2001, <https://www.rfc-editor.org/info/rfc2821>.

[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>.

[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>.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, DOI 10.17487/RFC4492, May 2006, <https://www.rfc-editor.org/info/rfc4492>.

[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>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

15.3. URIs

[1] https://en.wikipedia.org/wiki/Operational_Technology

[2] https://en.wikipedia.org/wiki/Single-root_input/output_virtualization

Appendix A. Background and Futures (Informative)

The following sections discuss additional background information about aspects of the normative parts of this document or associated mechanisms such as BRSKI (such as why specific choices were made by the ACP) and they provide discussion about possible future variations of the ACP.

A.1. ACP Address Space Schemes

This document defines the Zone, Vlong and Manual sub address schemes primarily to support address prefix assignment via distributed, potentially uncoordinated ACP registrars as defined in Section 6.10.7. This costs 48/46-bit identifier so that these ACP registrar can assign non-conflicting address prefixes. This design does not leave enough bits to simultaneously support a large number of nodes (Node-ID) plus a large prefix of local addresses for every node plus a large enough set of bits to identify a routing Zone. In result, Zone, Vlong 8/16 attempt to support all features, but in via separate prefixes.

In networks that always expect to rely on a centralized PMS as described above (Section 10.2.5), the 48/46-bits for the Registrar-ID could be saved. Such variations of the ACP addressing mechanisms could be introduced through future work in different ways. If the prefix rfcSELF in the ACP information field was changed, incompatible ACP variations could be created where every design aspect of the ACP could be changed. Including all addressing choices. If instead a new addressing sub-type would be defined, it could be a backward compatible extension of this ACP specification. Information such as the size of a zone-prefix and the length of the prefix assigned to the ACP node itself could be encoded via the extension field of the ACP domain information.

Note that an explicitly defined "Manual" addressing sub-scheme is always beneficial to provide an easy way for ACP nodes to prohibit incorrect manual configuration of any non-"Manual" ACP address spaces and therefore ensure that "Manual" operations will never impact correct routing for any non-"Manual" ACP addresses assigned via ACP domain certificates.

A.2. BRSKI Bootstrap (ANI)

[I-D.ietf-anima-bootstrapping-keyinfra] (BRSKI) describes how nodes with an IDevID certificate can securely and zero-touch enroll with a domain certificate (LDevID) to support the ACP. BRSKI also leverages the ACP to enable zero-touch bootstrap of new nodes across networks without any configuration requirements across the transit nodes (e.g., no DHCP/DNS forwarding/server setup). This includes otherwise not configured networks as described in Section 3.2. Therefore BRSKI in conjunction with ACP provides for a secure and zero-touch management solution for complete networks. Nodes supporting such an infrastructure (BRSKI and ACP) are called ANI nodes (Autonomic Networking Infrastructure), see [I-D.ietf-anima-reference-model]. Nodes that do not support an IDevID but only an (insecure) vendor specific Unique Device Identifier (UDI) or nodes whose manufacturer does not support a MASA could use some future security reduced version of BRSKI.

When BRSKI is used to provision a domain certificate (which is called enrollment), the BRSKI registrar (acting as an enhanced EST server) must include the subjectAltName / rfc822Name encoded ACP address and domain name to the enrolling node (called pledge) via its response to the pledges EST CSR Attribute request that is mandatory in BRSKI.

The Certificate Authority in an ACP network must not change the subjectAltName / rfc822Name in the certificate. The ACP nodes can therefore find their ACP address and domain using this field in the domain certificate, both for themselves, as well as for other nodes.

The use of BRSKI in conjunction with the ACP can also help to further simplify maintenance and renewal of domain certificates. Instead of relying on CRL, the lifetime of certificates can be made extremely small, for example in the order of hours. When a node fails to connect to the ACP within its certificate lifetime, it cannot connect to the ACP to renew its certificate across it (using just EST), but it can still renew its certificate as an "enrolled/expired pledge" via the BRSKI bootstrap proxy. This requires only that the BRSKI registrar honors expired domain certificates and that the pledge attempts to perform TLS authentication for BRSKI bootstrap using its expired domain certificate before falling back to attempting to use its IDevID for BRSKI. This mechanism could also render CRLs unnecessary because the BRSKI registrar in conjunction with the CA would not renew revoked certificates - only a "Do-not-renew" list would be necessary on BRSKI registrars/CA.

In the absence of BRSKI or less secure variants thereof, provisioning of certificates may involve one or more touches or non-standardized automation. Node vendors usually support provisioning of certificates into nodes via PKCS#7 (see [RFC2315]) and may support this provisioning through vendor specific models via Netconf ([RFC6241]). If such nodes also support Netconf Zero-Touch ([I-D.ietf-netconf-zerotouch]) then this can be combined to zero-touch provisioning of domain certificates into nodes. Unless there are equivalent integration of Netconf connections across the ACP as there is in BRSKI, this combination would not support zero-touch bootstrap across a not configured network though.

A.3. ACP Neighbor discovery protocol selection

This section discusses why GRASP DULL was chosen as the discovery protocol for L2 adjacent candidate ACP neighbors. The contenders considered where GRASP, mDNS or LLDP.

A.3.1. LLDP

LLDP and Cisco's earlier Cisco Discovery Protocol (CDP) are example of L2 discovery protocols that terminate their messages on L2 ports. If those protocols would be chosen for ACP neighbor discovery, ACP neighbor discovery would therefore also terminate on L2 ports. This would prevent ACP construction over non-ACP capable but LLDP or CDP enabled L2 switches. LLDP has extensions using different MAC addresses and this could have been an option for ACP discovery as well, but the additional required IEEE standardization and definition of a profile for such a modified instance of LLDP seemed to be more work than the benefit of "reusing the existing protocol" LLDP for this very simple purpose.

A.3.2. mDNS and L2 support

Multicast DNNS (mDNS) [RFC6762] with DNS Service Discovery (DNS-SD) Resource Records (RRs) as defined in [RFC6763] is a key contender as an ACP discovery protocol. because it relies on link-local IP multicast, it does operates at the subnet level, and is also found in L2 switches. The authors of this document are not aware of mDNS implementation that terminate their mDNS messages on L2 ports instead of the subnet level. If mDNS was used as the ACP discovery mechanism on an ACP capable (L3)/L2 switch as outlined in Section 7, then this would be necessary to implement. It is likely that termination of mDNS messages could only be applied to all mDNS messages from such a port, which would then make it necessary to software forward any non-ACP related mDNS messages to maintain prior non-ACP mDNS functionality. Adding support for ACP into such L2 switches with mDNS could therefore create regression problems for prior mDNS functionality on those nodes. With low performance of software forwarding in many L2 switches, this could also make the ACP risky to support on such L2 switches.

A.3.3. Why DULL GRASP

LLDP was not considered because of the above mentioned issues. mDNS was not selected because of the above L2 mDNS considerations and because of the following additional points:

If mDNS was not already existing in a node, it would be more work to implement than DULL GRASP, and if an existing implementation of mDNS was used, it would likely be more code space than a separate implementation of DULL GRASP or a shared implementation of DULL GRASP and GRASP in the ACP.

A.4. Choice of routing protocol (RPL)

This section motivates why RPL - "IPv6 Routing Protocol for Low-Power and Lossy Networks ([RFC6550] was chosen as the default (and in this specification only) routing protocol for the ACP. The choice and above explained profile was derived from a pre-standard implementation of ACP that was successfully deployed in operational networks.

Requirements for routing in the ACP are:

o Self-management: The ACP must build automatically, without human intervention. Therefore routing protocol must also work completely automatically. RPL is a simple, self-managing protocol, which does not require zones or areas; it is also self-configuring, since configuration is carried as part of the protocol (see Section 6.7.6 of [RFC6550]).

- Scale: The ACP builds over an entire domain, which could be a large enterprise or service provider network. The routing protocol must therefore support domains of 100,000 nodes or more, ideally without the need for zoning or separation into areas. RPL has this scale property. This is based on extensive use of default routing.

- Low resource consumption: The ACP supports traditional network infrastructure, thus runs in addition to traditional protocols. The ACP, and specifically the routing protocol must have low resource consumption both in terms of memory and CPU requirements. Specifically, at edge nodes, where memory and CPU are scarce, consumption should be minimal. RPL builds a destination-oriented directed acyclic graph (DODAG), where the main resource consumption is at the root of the DODAG. The closer to the edge of the network, the less state needs to be maintained. This adapts nicely to the typical network design. Also, all changes below a common parent node are kept below that parent node.

- Support for unstructured address space: In the Autonomic Networking Infrastructure, node addresses are identifiers, and may not be assigned in a topological way. Also, nodes may move topologically, without changing their address. Therefore, the routing protocol must support completely unstructured address space. RPL is specifically made for mobile ad-hoc networks, with no assumptions on topologically aligned addressing.

- Modularity: To keep the initial implementation small, yet allow later for more complex methods, it is highly desirable that the routing protocol has a simple base functionality, but can import new functional modules if needed. RPL has this property with the concept of "objective function", which is a plugin to modify routing behavior.

- Extensibility: Since the Autonomic Networking Infrastructure is a new concept, it is likely that changes in the way of operation will happen over time. RPL allows for new objective functions to be introduced later, which allow changes to the way the routing protocol creates the DAGs.

- Multi-topology support: It may become necessary in the future to support more than one DODAG for different purposes, using different objective functions. RPL allow for the creation of several parallel DODAGs, should this be required. This could be used to create different topologies to reach different roots.

o No need for path optimization: RPL does not necessarily compute the optimal path between any two nodes. However, the ACP does not require this today, since it carries mainly non-delay-sensitive feedback loops. It is possible that different optimization schemes become necessary in the future, but RPL can be expanded (see point "Extensibility" above).

A.5. ACP Information Distribution and multicast

IP multicast is not used by the ACP because the ANI (Autonomic Networking Infrastructure) itself does not require IP multicast but only service announcement/discovery. Using IP multicast for that would have made it necessary to develop a zero-touch auto configuring solution for ASM (Any Source Multicast - the original form of IP multicast defined in [RFC1112]), which would be quite complex and difficult to justify. One aspect of complexity where no attempt at a solution has been described in IETF documents is the automatic-selection of routers that should be PIM Sparse Mode (PIM-SM) Rendezvous Points (RPs) (see [RFC7761]). The other aspects of complexity are the implementation of MLD ([RFC4604]), PIM-SM and Anycast-RP (see [RFC4610]). If those implementations already exist in a product, then they would be very likely tied to accelerated forwarding which consumes hardware resources, and that in return is difficult to justify as a cost of performing only service discovery.

Some future ASA may need high performance in-network data replication. That is the case when the use of IP multicast is justified. Such an ASA can then use service discovery from ACP GRASP, and then they do not need ASM but only SSM (Source Specific Multicast, see [RFC4607]) for the IP multicast replication. SSM itself can simply be enabled in the Data-Plane (or even in an update to the ACP) without any other configuration than just enabling it on all nodes and only requires a simpler version of MLD (see [RFC5790]).

LSP (Link State Protocol) based IGP routing protocols typically have a mechanism to flood information, and such a mechanism could be used to flood GRASP objectives by defining them to be information of that IGP. This would be a possible optimization in future variations of the ACP that do use an LSP routing protocol. Note though that such a mechanism would not work easily for GRASP M_DISCOVERY messages which are intelligently (constrained) flooded not across the whole ACP, but only up to a node where a responder is found. We do expect that many future services in ASA will have only few consuming ASA, and for those cases, M_DISCOVERY is the more efficient method than flooding across the whole domain.

Because the ACP uses RPL, one desirable future extension is to use RPLs existing notion of loop-free distribution trees (DODAG) to make

GRASPs flooding more efficient both for M_FLOOD and M_DISCOVERY) See Section 6.12.5 how this will be specifically beneficial when using NBMA interfaces. This is not currently specified in this document because it is not quite clear yet what exactly the implications are to make GRASP flooding depend on RPL DODAG convergence and how difficult it would be to let GRASP flooding access the DODAG information.

A.6. Extending ACP channel negotiation (via GRASP)

[RFC Editor: This section to be removed before RFC.

[This section kept for informational purposes up until the last draft version as that would be the version that readers interested in the changelog would also go to to revisit it.]

The mechanism described in the normative part of this document to support multiple different ACP secure channel protocols without a single network wide MTI protocol is important to allow extending secure ACP channel protocols beyond what is specified in this document, but it will run into problem if it would be used for multiple protocols:

The need to potentially have multiple of these security associations even temporarily run in parallel to determine which of them works best does not support the most lightweight implementation options.

The simple policy of letting one side (Alice) decide what is best may not lead to the mutual best result.

The two limitations can easier be solved if the solution was more modular and as few as possible initial secure channel negotiation protocols would be used, and these protocols would then take on the responsibility to support more flexible objectives to negotiate the mutually preferred ACP security channel protocol.

IKEv2 is the IETF standard protocol to negotiate network security associations. It is meant to be extensible, but it is unclear whether it would be feasible to extend IKEv2 to support possible future requirements for ACP secure channel negotiation:

Consider the simple case where the use of native IPsec vs. IPsec via GRE is to be negotiated and the objective is the maximum throughput. Both sides would indicate some agreed upon performance metric and the preferred encapsulation is the one with the higher performance of the slower side. IKEv2 does not support negotiation with this objective.

Consider DTLS and some form of MacSec are to be added as negotiation options - and the performance objective should work across all IPsec, DTLS and MacSec options. In the case of MacSEC, the negotiation would also need to determine a key for the peering. It is unclear if it would be even appropriate to consider extending the scope of negotiation in IKEv2 to those cases. Even if feasible to define, it is unclear if implementations of IKEv2 would be eager to adopt those type of extension given the long cycles of security testing that necessarily goes along with core security protocols such as IKEv2 implementations.

A more modular alternative to extending IKEv2 could be to layer a modular negotiation mechanism on top of the multitude of existing or possible future secure channel protocols. For this, GRASP over TLS could be considered as a first ACP secure channel negotiation protocol. The following are initial considerations for such an approach. A full specification is subject to a separate document:

To explicitly allow negotiation of the ACP channel protocol, GRASP over a TLS connection using the GRASP_LISTEN_PORT and the node's and peer's link-local IPv6 address is used. When Alice and Bob support GRASP negotiation, they do prefer it over any other non-explicitly negotiated security association protocol and should wait trying any non-negotiated ACP channel protocol until after it is clear that GRASP/TLS will not work to the peer.

When Alice and Bob successfully establish the GRASP/TSL session, they will negotiate the channel mechanism to use using objectives such as performance and perceived quality of the security. After agreeing on a channel mechanism, Alice and Bob start the selected Channel protocol. Once the secure channel protocol is successfully running, the GRASP/TLS connection can be kept alive or timed out as long as the selected channel protocol has a secure association between Alice and Bob. When it terminates, it needs to be re-negotiated via GRASP/TLS.

Notes:

- o Negotiation of a channel type may require IANA assignments of code points.

- o TLS is subject to reset attacks, which IKEv2 is not. Normally, ACP connections (as specified in this document) will be over link-local addresses so the attack surface for this one issue in TCP should be reduced (note that this may not be true when ACP is tunneled as described in Section 8.2.2.

o GRASP packets received inside a TLS connection established for GRASP/TLS ACP negotiation are assigned to a separate GRASP domain unique to that TLS connection.

A.7. CAs, domains and routing subdomains

There is a wide range of setting up different ACP solution by appropriately using CAs and the domain and rsub elements in the domain information field of the domain certificate. We summarize these options here as they have been explained in different parts of the document in before and discuss possible and desirable extensions:

An ACP domain is the set of all ACP nodes using certificates from one or more Trust Anchors (TA) (typically one CA) using the same domain field. GRASP inside the ACP is run across all transitively connected ACP nodes in a domain.

The rsub element in the domain information field permits the use of addresses from different ULA prefixes. One use case is to create multiple physical networks that initially may be separated with one ACP domain but different routing subdomains, so that all nodes can mutual trust their ACP domain certificates (not depending on rsub) and so that they could connect later together into a contiguous ACP network.

One instance of such a use case is an ACP for regions interconnected via a non-ACP enabled core, for example due to the absence of product support for ACP on the core nodes. ACP connect configurations as defined in this document can be used to extend and interconnect those ACP islands to the NOC and merge them into a single ACP when later that product support gap is closed.

Note that RPL scales very well. It is not necessary to use multiple routing subdomains to scale ACP domains in a way it would be possible if other routing protocols where used. They exist only as options for the above mentioned reasons.

If different ACP domains are to be created that should not allow to connect to each other by default, these ACP domains simply need to have different domain elements in the domain information field. These domain elements can be arbitrary, including subdomains of one another: Domains "example.com" and "research.example.com" are separate domains if both are domain elements in the domain information element of certificates.

It is not necessary to have a separate CA for different ACP domains: an operator can use a single CA to sign certificates for multiple ACP domains that are not allowed to connect to each other because the checks for ACP adjacencies includes comparison of the domain part.

If multiple independent networks choose the same domain name but had their own CA, these would not form a single ACP domain because of CA mismatch. Therefore there is no problem in choosing domain names that are potentially also used by others. Nevertheless it is highly recommended to use domain names that one can have high probability to be unique. It is recommended to use domain names that start with a DNS domain names owned by the assigning organization and unique within it. For example "acp.example.com" if you own "example.com".

A.8. Intent for the ACP

Intent is the architecture component of autonomic networks according to [I-D.ietf-anima-reference-model] that allows operators to issue policies to the network. 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 trust anchors (CA) and domain names to the ACP domain membership check (Section 6.1.3) 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.9. Adopting ACP concepts for other environments

The ACP as specified in this document is very explicit about the choice of options to allow interoperable implementations. The choices made may not be the best for all environments, but the concepts used by the ACP can be used to build derived solutions:

The ACP specifies the use of ULA and deriving its prefix from the domain name so that no address allocation is required to deploy the ACP. The ACP will equally work not using ULA but any other /48 IPv6 prefix. This prefix could simply be a configuration of the ACP registrars (for example when using BRSKI) to enroll the domain certificates - instead of the ACP registrar deriving the /48 ULA prefix from the AN domain name.

Some solutions may already have an auto-addressing scheme, for example derived from existing unique device identifiers (e.g., MAC addresses). In those cases it may not be desirable to assign addresses to devices via the ACP address information field in the way described in this document. The certificate may simply serve to identify the ACP domain, and the address field could be empty/unused. The only fix required in the remaining way the ACP operate is to define another element in the domain certificate for the two peers to decide who is Alice and who is Bob during secure channel building. Note though that future work may leverage the acp address to authenticate "ownership" of the address by the device. If the address used by a device is derived from some pre-existing permanent local ID (such as MAC address), then it would be useful to store that address in the certificate using the format of the access address information field or in a similar way.

The ACP is defined as a separate VRF because it intends to support well managed networks with a wide variety of configurations. Therefore, reliable, configuration-indestructible connectivity cannot be achieved from the Data-Plane itself. In solutions where all transit connectivity impacting functions are fully automated (including security), indestructible and resilient, it would be possible to eliminate the need for the ACP to be a separate VRF. Consider the most simple example system in which there is no separate Data-Plane, but the ACP is the Data-Plane. Add BRSKI, and it becomes a fully autonomic network - except that it does not support automatic addressing for user equipment. This gap can then be closed for example by adding a solution derived from [I-D.ietf-anima-prefix-management].

TCP/TLS as the protocols to provide reliability and security to GRASP in the ACP may not be the preferred choice in constrained networks. For example, CoAP/DTLS (Constrained Application Protocol) may be preferred where they are already used, allowing to reduce the additional code space footprint for the ACP on those devices. Hop-by-hop reliability for ACP GRASP messages could be made to support protocols like DTLS by adding the same type of negotiation as defined in this document for ACP secure channel protocol negotiation. End-to-end GRASP connections can be made to select their transport protocol in future extensions of the ACP meant to better support constrained devices by indicating the supported transport protocols (e.g.: TLS/DTLS) via GRASP parameters of the GRASP objective through which the transport endpoint is discovered.

The routing protocol chosen by the ACP design (RPL) does explicitly not optimize for shortest paths and fastest convergence. Variations of the ACP may want to use a different routing protocol or introduce more advanced RPL profiles.

Variations such as what routing protocol to use, or whether to instantiate an ACP in a VRF or (as suggested above) as the actual Data-Plane, can be automatically chosen in implementations built to support multiple options by deriving them from future parameters in the certificate. Parameters in certificates should be limited to those that would not need to be changed more often than certificates would need to be updated anyhow; Or by ensuring that these parameters can be provisioned before the variation of an ACP is activated in a node. Using BRSKI, this could be done for example as additional follow-up signaling directly after the certificate enrollment, still leveraging the BRSKI TLS connection and therefore not introducing any additional connectivity requirements.

Last but not least, secure channel protocols including their encapsulations are easily added to ACP solutions. ACP hop-by-hop network layer secure channels could also be replaced by end-to-end security plus other means for infrastructure protection. Any future network OAM should always use end-to-end security anyhow and can leverage the domain certificates and is therefore not dependent on security to be provided for by ACP secure channels.

A.10. Further (future) options

A.10.1. Auto-aggregation of routes

Routing in the ACP according to this specification only leverages the standard RPL mechanism of route optimization, e.g. keeping only routes that are not towards the RPL root. This is known to scale to networks with 20,000 or more nodes. There is no auto-aggregation of routes for /48 ULA prefixes (when using rsub in the domain information field) and/or Zone-ID based prefixes.

Automatic assignment of Zone-ID and auto-aggregation of routes could be achieved for example by configuring zone-boundaries, announcing via GRASP into the zones the zone parameters (zone-ID and /48 ULA prefix) and auto-aggegating routes on the zone-boundaries. Nodes would assign their Zone-ID and potentially even /48 prefix based on the GRASP announcements.

A.10.2. More options for avoiding IPv6 Data-Plane dependency

As described in Section 6.12.2, the ACP depends on the Data-Plane to establish IPv6 link-local addressing on interfaces. Using a separate MAC address for the ACP allows to fully isolate the ACP from the data-plane in a way that is compatible with this specification. It is also an ideal option when using Single-root input/output virtualization (SR-IOV - see https://en.wikipedia.org/wiki/Single-root_input/output_virtualization [2]) in an implementation to isolate the ACP because different SR-IOV interfaces use different MAC addresses.

When additional MAC address(es) are not available, separation of the ACP could be done at different demux points. The same subnet interface could have a separate IPv6 interface for the ACP and Data-Plane and therefore separate link-local addresses for both, where the ACP interface is non-configurable on the Data-Plane. This too would be compatible with this specification and not impact interoperability.

An option that would require additional specification is to use a different Ethertype from 0x86DD (IPv6) to encapsulate IPv6 packets for the ACP. This would be a similar approach as used for IP authentication packets in [IEEE-802.1X] which use the Extensible Authentication Protocol over Local Area Network (EAPoL) ethertype (0x88A2).

Note that in the case of ANI nodes, all the above considerations equally apply to the encapsulation of BRSKI packets including GRASP used for BRSKI.

A.10.3. ACP APIs and operational models (YANG)

Future work should define YANG ([RFC7950]) data model and/or node internal APIs to monitor and manage the ACP.

Support for the ACP Adjacency Table (Section 6.2) and ACP GRASP need to be included into such model/API.

A.10.4. RPL enhancements

```
      ..... USA ......              ..... Europe ......

           NOC1                           NOC2
            |                               |
            |           metric 100          |
          ACP1 --------------------------- ACP2   .
            |                               |     . WAN
            | metric 10          metric 20  |     . Core
            |                               |     .
          ACP3 --------------------------- ACP4   .
            |           metric 100          |
            |                               |     .
            |                               |     . Sites
          ACP10                           ACP11   .
```

Figure 18: Dual NOC

The profile for RPL specified in this document builds only one spanning-tree path set to a root (NOC). In the presence of multiple NOCs, routing toward the non-root NOCs may be suboptimal. Figure 18 shows an extreme example. Assuming that node ACP1 becomes the RPL root, traffic between ACP11 and NOC2 will pass through ACP4-ACP3-ACP1-ACP2 instead of ACP4-ACP2 because the RPL calculated DODAG/routes are shortest paths towards the RPL root.

To overcome these limitations, extensions/modifications to the RPL profile can provide optimality for multiple NOCs. This requires utilizing Data-Plane artifact including IPinIP encap/decap on ACP routers and processing of IPv6 RPI headers. Alternatively, (Src,Dst) routing table entries could be used.

Flooding of ACP GRASP messages can be further constrained and therefore optimized by flooding only via links that are part of the RPL DODAG.

A.10.5. Role assignments

ACP connect is an explicit mechanism to "leak" ACP traffic explicitly (for example in a NOC). It is therefore also a possible security gap when it is easy to enable ACP connect on arbitrary compromised ACP nodes.

One simple solution is to define an extension in the ACP certificates ACP information field indicating the permission for ACP connect to be configured on that ACP node. This could similarly be done to decide whether a node is permitted to be a registrar or not.

Tying the permitted "roles" of an ACP node to the ACP domain certificate provides fairly strong protection against misconfiguration, but is still subject to code modifications.

Another interesting role to assign to certificates is that of a NOC node. This would allow to limit certain type of connections such as OAM TLS connections to only NOC initiator or responders.

A.10.6. Autonomic L3 transit

In this specification, the ACP can only establish autonomic connectivity across L2 hops and only explicitly configured options to tunnel across L3. Future work should specify mechanisms to automatically tunnel ACP across L3 networks. A hub&spoke option would allow to tunnel across the Internet to a cloud or central instance of the ACP, a peer-to-peer tunneling mechanism could tunnel ACP islands across an L3VPN infrastructure.

A.10.7. Diagnostics

Section 10.1 describes diagnostics options that can be done without changing the external, interoperability affecting characteristics of ACP implementations.

Even better diagnostics of ACP operations is possible with additional signaling extensions, such as:

1. Consider if LLDP should be a recommended functionality for ANI devices to improve diagnostics, and if so, which information elements it should signal (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 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.10.8. Avoiding and dealing with compromised ACP nodes

Compromised ACP nodes pose the biggest risk to the operations of the network. The most common type of compromise is leakage of credentials to manage/configure the device and the application of malicious configuration including the change of access credentials, but not the change of software. Most of todays networking equipment should have secure boot/software infrastructure anyhow, so attacks that introduce malicious software should be a lot harder.

The most important aspect of security design against these type of attacks is to eliminate password based configuration access methods and instead rely on certificate based credentials handed out only to nodes where it is clear that the private keys can not leak. This limits unexpected propagation of credentials.

If password based credentials to configure devices still need to be supported, they must not be locally configurable, but only be remotely provisioned or verified (through protocols like Radius or Diameter), and there must be no local configuration permitting to change these authentication mechanisms, but ideally they should be autoconfiguring across the ACP. See [I-D.eckert-anima-noc-autoconfig].

Without physical access to the compromised device, attackers with access to configuration should not be able to break the ACP connectivity, even when they can break or otherwise manipulate (spoof) the data-plane connectivity through configuration. To achieve this, it is necessary to avoid providing configuration options for the ACP, such as enabling/disabling it on interfaces. For example there could be an ACP configuration that locks down the current ACP config unless factory reseet is done.

With such means, the valid administration has the best chances to maintain access to ACP nodes, discover malicious configuration though ongoing configuration tracking from central locations for example, and to react accordingly.

The primary reaction is withdrawal/change of credentials, terminate malicious existing management sessions and fixing the configuration. Ensuring that management sessions using invalidated credentials are terminated automatically without recourse will likely require new work.

Only when these steps are not feasible would it be necessary to revoke or expire the ACP domain certificate credentials and consider the node kicked off the network - until the situation can be further rectified, likely requiring direct physical access to the node.

Without extensions, compromised ACP nodes can only be removed from the ACP at the speed of CRL/OCSP information refresh or expiry (and non-removal) of short lived certificates. Future extensions to the ACP could for example use GRASP flooding distribution of triggered updates of CRL/OCSP or explicit removal indication of the compromised nodes domain certificate.

Authors' Addresses

Toerless Eckert (editor)
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara 95050
USA

Email: tte+ietf@cs.fau.de

Michael H. Behringer (editor)

Email: michael.h.behringer@gmail.com

Steinthor Bjarnason
Arbor Networks
2727 South State Street, Suite 200
Ann Arbor MI 48104
United States

Email: sbjarnason@arbor.net

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2020

M. Pritikin
Cisco
M. Richardson
Sandelman
T. Eckert
Futurewei USA
M. Behringer

K. Watsen
Watsen Networks
October 28, 2019

Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-29

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 will expire on April 30, 2020.

Table of Contents

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] IDevID) 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):

- o using the Implicit Trust Anchor [RFC7030] database to authenticate an owner specific service (not an autonomic solution because the URL must be securely distributed),
- o engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- o using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),
- o 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:

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

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

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in 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]. This document 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 [IDevID]

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: A 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 accurately 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:

- o Device management.

- o Routing authentication.

- o 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.

```
                                              +------------------------+
      +--------------Drop Ship----------------| Vendor Service         |
      |                                       +------------------------+
      |                                       | M anufacturer|         |
      |                                       | A uthorized  |Ownership|
      |                                       | S igning     |Tracker  |
      |                                       | A uthority   |         |
      |                                       +--------------+---------+
      |                                                      ^
      |                                                      |  BRSKI-
      V                                                      |   MASA
   +-------+     ............................................|...
   |       |     .                                           |  .
   |       |     .  +------------+       +-----------+       |  .
   |       |     .  |            |       |           |       |  .
   |Pledge |     .  |   Join     |       | Domain    <-------+  .
   |       |     .  |   Proxy    |       | Registrar |          .
   |       <-------->............<-------> (PKI RA)  |          .
   |       |        |        BRSKI-EST   |           |          .
   |       |     .  |            |       +-----+-----+          .
   |IDevID |     .  +------------+             | e.g. RFC7030   .
   |       |     .           +-----------------+----------+     .
   |       |     .           | Key Infrastructure         |     .
   |       |     .           | (e.g., PKI Certificate     |     .
   +-------+     .           |       Authority)           |     .
                 .           +----------------------------+     .
                 .                                              .
                 ................................................
                               "Domain" 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.

```
                  ------------
                 /  Factory   \
                 \  default   /
                  -----+------
                       |
                +------v-------+
                | (1) Discover |
   +------------>              |
   |            +------+-------+
   |                   |
   |            +------v-------+
   |            | (2) Identify |
   ^------------+              |
   | rejected   +------+-------+
   |                   |
   |            +------v-------+
   |            | (3) Request  |
   |            |     Join     |
   |            +------+-------+
   |                   |
   |            +------v-------+
   |            | (4) Imprint  |
   ^------------+              |
   | Bad MASA   +------+-------+
   | response          |  send Voucher Status Telemetry
   |            +------v-------+
   |            | (5) Enroll   |<---+ (non-error HTTP codes  )
   ^------------+              |\___/ (e.g. 202 'Retry-After')
   | Enroll     +------+-------+
   | Failure           |
   |              -----v------
   |             /  Enrolled  \
   ^------------+             |
    Factory      \------------/
    reset
```

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).

The serialNumber fields is defined in [RFC5280]. That specification allows for the field to be omitted if there is a good technical reason. IDevID certificates for use with this protocol are REQUIRED to include the "serialNumber" attribute with the device's unique serial number (from [IDevID] section 7.2.8, and [RFC5280] section 4.1.2.2's list of standard attributes).

The serialNumber field 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/est" 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

```
   +--------+         +---------+    +------------+     +------------+
   | Pledge |         | Circuit |    | Domain     |     | Vendor     |
   |        |         | Join    |    | Registrar  |     | Service    |
   |        |         | Proxy   |    |  (JRC)     |     | (MASA)     |
   +--------+         +---------+    +------------+     +------------+
     |                     |                   |           Internet |
   [discover]              |                   |                    |
     |<-RFC4862 IPv6 addr  |                   |                    |
     |<-RFC3927 IPv4 addr  | Appendix A        |  Legend            |
     |-++++++++++++++++++->|                   |  C - circuit       |
     | optional: mDNS query| Appendix B        |      join proxy    |
     | RFC6763/RFC6762 (+) |                   |  P - provisional   |
     |<-++++++++++++++++++-|                   |    TLS connection  |
     | GRASP M_FLOOD       |                   |                    |
     |   periodic broadcast|                   |                    |
   [identity]              |                   |                    |
     |<------------------->C<----------------->|                    |
     |         TLS via the Join Proxy          |                    |
     |<--Registrar TLS server authentication---|                    |
   [PROVISIONAL accept of server cert]         |                    |
     P---X.509 client authentication---------->|                    |
   [request join]                              |                    |
     P---Voucher Request(w/nonce for voucher)->|                    |
     P                  /-------------------   |                    |
     P                  |                 [accept device?]          |
     P                  |                 [contact Vendor]          |
     P                  |                      |--Pledge ID-------->|
     P                  |                      |--Domain ID-------->|
     P                  |                      |--optional:nonce--->|
     P              optional:                  |     [extract DomainID]
     P        can occur in advance             |     [update audit log]
     P            if nonceleess                |                    |
     P                  |                      |<- voucher ---------|
     P                  \-------------------   | w/nonce if provided|
     P<------voucher---------------------------|                    |
   [imprint]                                   |                    |
     |-------voucher status telemetry--------->|                    |
     |                                         |<-device audit log--|
     |                             [verify audit log and voucher]   |
     |<--------------------------------------->|                    |
   [enroll]                                    |                    |
     | Continue with RFC7030 enrollment        |                    |
     | using now bidirectionally authenticated |                    |
     | TLS session.                            |                    |
   [enrolled]                                  |                    |
```

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 can 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 in, MUST check the above time fields in all certificates and signatures that ir 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 they 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.

If the registrar is integrated with [RFC8520] and the pledge IDevID contains the id-pe-mud-url then the registrar MAY attempt to obtain the MASA URL from the MUD file. The MUD file extension for the MASA URL is defined in Appendix C.

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

The registrar in turn forms the "registrar voucher-request", signs it with it's 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 by base64 encoded ([RFC4648]). The contents of the certificate have been elided to save space. For detailed examples, see Appendix D.2. These examples conform to the encoding rules defined in [RFC7951].

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


```
{
    "ietf-voucher-request:voucher": {
        "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 Profile 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 RFC2119 RFC8174 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: Voucher Profile for Bootstrapping Protocols";
  }
```

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

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

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

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

      refine "voucher/domain-cert-revocation-checks" {
        description "The domain-cert-revocation-checks field
                     is not valid in a voucher request, and
                     any occurence 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-T X.690.

             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 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 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:

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

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

ttl             = 180000     ; 180,000 ms (3 minutes)
initiator = ACP address to contact Registrar
objective-flags   = sync-only  ; as in GRASP spec
sync-only         =  4         ; M_FLOOD only requires synchronization
loop-count        =  1         ; one hop only
objective-value   =  any       ; none

locator-option    = [ O_IPv6_LOCATOR, ipv6-address,
                    transport-proto, port-number ]
ipv6-address      = the v6 LL of the Proxy
$transport-proto /= IPPROTO_TCP   ; note this can be any value from the
                                 ; IANA protocol registry, as per
                                 ; [GRASP] section 2.9.5.1, note 3.
port-number      = selected by Proxy
```

Figure 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 it's own announcement.

4.2. CoAP connection to Registrar

The use of CoAP to connect from pledge to registrar is out of scope for this document, and is described in future work. See [I-D.ietf-anima-constrained-voucher].

4.3. Proxy discovery and communication of Registrar

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

The registrar announces itself using ACP instance of GRASP using M_FLOOD messages. 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, 12340815, 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:

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

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

initiator = ACP address to contact Registrar
objective-flags = sync-only  ; as in GRASP spec
sync-only =  4               ; M_FLOOD only requires synchronization
loop-count      = 255        ; mandatory maximum
objective-value = text       ; name of the (list of) of supported
                             ; protocols: "EST-TLS" for RFC7030.
```

Figure 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 same "/.well-known" URI tree as the existing EST URIs as described in EST [RFC7030] section 3.2.2. The communication channel between the pledge and the registrar is referred to as "BRSKI-EST" (see Figure 1).

The communication channel between the registrar and MASA is similarly described as extensions to EST within the same "/.well-known" tree. For clarity this channel is referred to as "BRSKI-MASA". (See Figure 1).

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

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:

- o 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 signatures 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/est/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 generated across multiple boots, or after a factory reset. Difference nonces MUST NOT 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, and 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:

- o allow any device of a specific type (as determined by the X.509 IDevID),
- o allow any device from a specific vendor (as determined by the X.509 IDevID),
- o 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 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 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 or Digest 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/est/requestvoucher".

The voucher media type "application/voucher-cms+json" is defined in [RFC8366] and is also used for the registrar voucher-request. It is a JSON document that has been signed using a CMS structure. The registrar MUST sign the registrar voucher-request. The entire registrar certificate chain, up to and including the Domain CA, MUST be included in the CMS structure.

MASA impementations SHOULD anticipate future media types but of course will simply fail the request if those types are not yet known.

The Registrar SHOULD include an [RFC7231] section 5.3.2 "Accept" header 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. MASA's are expected to be flexible in what they accept.

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

The registrar's certificate chain is extracted from the signature method. The entire registrar certificate chain was included in the CMS structure, as specified in Section 5.5. This CA certificate will be used to populate the "pinned-domain-cert" of the voucher being issued.

If this domain 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.

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 cmcRC 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 (blinked LED etc) during this wait cycle if mechanisms for this are available. To prevent an attacker registrar from significantly delaying bootstrapping the pledge MUST limit the 'Retry-After' time to 60 seconds. 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 origins. EST supports redirection but requires user input; this change allows the pledge to follow a single redirection without a user interaction.

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

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

A 406 (Not Acceptable) response is appropriate if a voucher of the desired type or using the desired algorithms (as indicated by the

Accept: 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: The domain CA cert. See Section 5.5.2. This figure is illustrative, for an example, see Appendix D.2

serial-number: The serial-number as provided in the voucher-request. Also see Section 5.5.5.

domain-cert-revocation-checks: Set as appropriate for the pledge's capabilities and as documented in [RFC8366]. The MASA MAY set this field to 'false' since setting it to 'true' would require that revocation information be available to the pledge and this document does not make normative requirements for [RFC6961] or equivalent integrations.

expires-on: This is set for nonceless vouchers. The MASA ensures the voucher lifetime is consistent with any revocation or pinned-domain-cert consistency checks the pledge might perform. See section Section 2.6.1. There are three times to consider: (a) a configured voucher lifetime in the MASA, (b) the expiry time for the registrar's certificate, (c) any certificate revocation information (CRL) lifetime. The expires-on field SHOULD be before the earliest of these three values. Typically (b) will be some significant time in the future, but (c) will typically be short (on the order of a week or less). The RECOMMENDED period for (a) is on the order of 20 minutes, so it will typically determine the lifespan of the resulting voucher. 20 minutes is sufficient time to reach the post-provisional state in the pledge, at which point there is an established trust relationship between pledge and registrar. The subsequent operations can take as long as required from that point onwards. The lifetime of the voucher has no impact on the lifespan of the ownership relationship.

Whenever a voucher is issued the MASA MUST update the audit-log 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

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

If a registrar's credentials cannot be verified using the pinned-domain-cert trust anchor from the voucher then the TLS connection is immediately discarded and the pledge abandons attempts to bootstrap with this discovered registrar. The pledge SHOULD send voucher status telemetry (described below) before closing the TLS connection. The pledge MUST attempt to enroll using any other proxies it has found. It SHOULD return to the same proxy again after 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 an trust anchor for future operations such as enrollment (e.g. [RFC7030] as recommended) or trust anchor management or raw protocols that do not need full PKI based key management. It can be used to authenticate any dynamically discovered EST server that contain the id-kp-cmcRA extended key usage extension as detailed in EST RFC7030 section 3.6.1; but to reduce system complexity the pledge SHOULD avoid additional discovery operations. Instead the pledge SHOULD communicate directly with the registrar as the EST server. The 'pinned-domain-cert' is not a complete distribution of the [RFC7030] section 4.1.3 CA Certificate Response, which is an additional justification for the recommendation to proceed with EST key management operations. Once a full CA Certificate Response is obtained it is more authoritative for the domain than the limited 'pinned-domain-cert' response.

5.7. Pledge 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.

To indicate pledge status regarding the voucher, the pledge MUST post 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/est/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 accesptable.

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.

The keys to this JSON object are case-sensitive and MUST be lowercase. Figure 15 shows an example JSON.

```
{
    "version":"1",
    "status":false,
    "reason":"Informative human readable message",
    "reason-context": { "additional" : "JSON" }
}
```


Figure 15: 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 8.4. 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/est/requestauditlog".

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

A registrar MAY request logs at future times. If the registrar generates a new request then the MASA is forced to perform the additional cryptographic operations to verify the new request.

A MASA that receives a request for a device that does not exist, or for which the requesting owner was never an owner returns an HTTP 404 ("Not found") code.

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 occurance. 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 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:

```
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,
}
```

Figure 16: CDDL for audit-log response

As an abstract 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":"f4G6Vi1t8nKo/FieCVgpBg==",
        "assertion":"proximity"
    }
  ],
    "truncation": {
        "nonced duplicates": "0",
        "nonceless duplicates": "1",
        "arbitrary": "2"
     }
}
```


Figure 17: 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 the 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 but this can still be problematic for registrar's that expected only new (not pre-owned) pledges. A "logged" assertion informs the registrar that the prior vouchers were issued with minimal verification. A "proximity" assertion assures the registrar that the pledge was truly communicating with the prior domain and thus provides assurance that the prior domain really has deployed the pledge.

A relatively simple policy is to white list known (internal or external) domainIDs, and require all vouchers to have a nonce. An alternative is to require that all nonceless vouchers 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.

In order to communicate this indicator, the client HTTP POSTs the following to the server at the new EST endpoint at "/.well-known/est/enrollstatus".

To indicate 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 close the existing TLS connection, and start a new one.

In the case of a FAIL, the Reason string indicates why the most recent enrollment failed.

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.

In the case of a SUCCESS the Reason string is omitted.

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 18: 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/est/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:

```
+--------+         +---------+    +------------+     +------------+
| Pledge |         | Join    |    | Domain     |     |Manufacturer|
|        |         | Proxy   |    | Registrar  |     | Service    |
|        |         |         |    |            |     | (Internet) |
+--------+         +---------+    +------------+     +------------+
```

Figure 10

Pledge: The pledge could be compromised and providing an attack
   vector for malware. The entity is trusted to only imprint using

secure methods described in this document. Additional endpoint assessment techniques are RECOMMENDED but are out-of-scope of this document.

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

Registrar: When interacting with a MASA a registrar makes all decisions. For Ownership Audit Vouchers (see [RFC8366]) the registrar is provided an opportunity to accept MASA decisions.

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

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 has registered the following:

```
URI: urn:ietf:params:xml:ns:yang:ietf-mud-brski-masa
Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

8.2. Well-known EST registration

This document extends the definitions of "est" (so far defined via RFC7030) in the "https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml" registry. IANA is asked to change the registration of "est" to include RFC7030 and this document.

8.3. 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.

8.4. Pledge BRSKI Status Telemetry

IANA is requested to create a new Registry entitled: "BRSKI Parameters", and within that Registry to create a table called: "Pledge BRSKI Status Telemetry Attributes". New items can be added using the Specification Required. The following items are to be in the initial registration, with this document (Section 5.7) as the reference:

- version
- Status
- Reason
- reason-context

8.5. 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]

8.6. MUD File Extension for the MASA

The IANA is requested to list the name "masa" in the MUD extensions registry defined in [RFC8520]. Its use is documented in Appendix C.

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 asssertion 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 significiant 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 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 Certicate pinned, or HTTP Authentication process. The MASA creates signed voucher artifacts according to a it’s 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 seperate 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 approriate 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 (MTIM) 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

The so-called "call-home" mechanism that occurs as part of the BRSKI-MASA connection standardizes what has been deemed by some as a sinister mechanism for corporate oversight of individuals. ([livingwithIoT] and [IoTstrangeThings] for a small sample).

As the Autonomic Control Plane (ACP) usage of BRSKI is not targeted at individual usage of IoT devices, but rather at the Enterprise and ISP creation of networks in a zero-touch fashion, the "call-home" represents a different kind of concern.

It needs to be re-iterated that the BRSKI-MASA mechanism only occurs once during the commissioning of the device. It is well defined, and although encrypted with TLS, it could in theory be made auditable as the contents are well defined. This connection does not occur when the device powers on or is restarted for normal routines. (It is conceivable, 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 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:

- o 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.

- o 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.

- o the time the device is activated,

- o 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. Availability of good random numbers

Although the nonce used by the Pledge in the voucher-request does not require a strong cryptographic randomness, the use of TLS in all of the protocols in this document does.

In particular implementations should pay attention to the advance in [RFC4086] section 3, particularly section 3.4. Devices which are reset to factory default in order to perform a second bootstrap with a new owner MUST NOT seed their random number generators in the same way across bootstraps.

11.3. 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.

11.4. 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:

- o Manually configuring each manufacturer's trust anchor.

- o 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.

- o scanning the trust anchor from a QR code that came with the packaging (this is really a manual TOFU mechanism)

- o 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.

- o 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.

11.5. 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.

11.5.1. 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.

11.5.2. 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.

11.5.2.1. Attacker opportunties 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 consumers 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.

11.5.2.2. 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 that 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.

11.5.3. 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.

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, 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)", draft-ietf-anima-autonomic-control-plane-20 (work in progress), July 2019.

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

[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., "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>.

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

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

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

[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>.

[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>.

[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>.

[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)", draft-ietf-ace-coap-est-15 (work in progress), October 2019.

[I-D.ietf-anima-constrained-voucher] Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols", draft-ietf-anima-constrained-voucher-05 (work in progress), July 2019.

[I-D.ietf-anima-reference-model] Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-10 (work in progress), November 2018.

[I-D.ietf-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-ietf-anima-stable-connectivity-10 (work in progress), February 2018.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", draft-ietf-netconf-keystore-12 (work in progress), July 2019.

[I-D.richardson-anima-state-for-joinrouter]
Richardson, M., "Considerations for stateful vs stateless join router in ANIMA bootstrap", draft-richardson-anima-state-for-joinrouter-02 (work in progress), January 2018.

[imprinting]
"Wikipedia article: Imprinting", July 2015, <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>.

[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>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

[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>.

[slowloris]
"Slowloris (computer security)", February 2019, <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, February 2014, <http://www.w3.org/TR/2014/WD-capability-urls-20140218>.

Appendix A. IPv4 and non-ANI operations

The secification of BRSKI in Section 4 intentionally only covers the mechanisms for an IPv6 pledge using Link-Local addresses. This section describes non-normative extensions that can be used in other environments.

A.1. IPv4 Link Local addresses

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

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

A.2. Use of DHCPv4

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

Appendix B. mDNS / DNSSD proxy discovery options

Pledge discovery of the proxy (Section 4.1) MAY be performed with DNS-based Service Discovery [RFC6763] over Multicast DNS [RFC6762] to discover the proxy at "_brski-proxy._tcp.local.".

Proxy discovery of the registrar (Section 4.3) MAY be performed with DNS-based Service Discovery over Multicast DNS to discover registrars by searching for the service "_brski-registrar._tcp.local.".

To prevent unaccceptable 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.<domain>". 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. MUD Extension

The following extension augments the MUD model to include a single node, as described in [RFC8520] section 3.6, using the following sample module that has the following tree structure:

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

The model is defined as follows:

```
<CODE BEGINS> file "ietf-mud-brski-masaurl-extension@2018-02-14.yang"
module ietf-mud-brski-masa {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-brski-masa";
  prefix ietf-mud-brski-masa;
  import ietf-mud {
    prefix ietf-mud;
  }
  import ietf-inet-types {
    prefix inet;
  }

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

  revision 2018-02-14 {
    description
    "Initial revision.";
    reference
    "RFC XXXX: Manufacturer Usage Description
    Specification";
  }

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

The MUD extensions string "masa" is defined, and MUST be included in the extensions array of the mud container of a MUD file when this extension is used.

Appendix D. Example Vouchers

Three entities are involved in a voucher: the MASA issues (signs) it, the registrar's public key is mentioned in the voucher, and the pledge validates it. In order to provide reproduceable examples the public and private keys for an example MASA and registrar are first listed.

D.1. Keys involved

The Manufacturer has a Certificate Authority that signs the pledge's IDevID. In addition the Manufacturer's signing authority (the MASA) signs the vouchers, and that certificate must distributed to the devices at manufacturing time so that vouchers can be validated.

D.1.1. MASA key pair for voucher signatures

This private key signs vouchers:


```
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAgiRoYqKoEcfOfvRvmZ5P5Azn58tuI7nSnIy7OgFnCeiNo+BmbgMho
r6lcU60gwVagBwYFK4EEACKhZANiAATZAH3Rb2FvIJOnts+vXuWW35ofyNbCHzjA
zOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCfw5ICgJ8CuM3vV5ty9bf7KUlOkejz
Tvv+5PV++elkP9HQ83vqTAws2WwWTxI=
-----END EC PRIVATE KEY-----
```


This public key validates vouchers:


```
-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQQDAjBNMRIwEAYKCZImiZPyLGQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
IEhpZ2h3YXkgQ0EwHhcNMTcwMzI2MTYxOTQwWhcNMTkwMzI2MTYxOTQwWjBHMRIw
EAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xFjAU
BgNVBAMMDVVuc3RydW5nIE1BU0EwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAATZAH3R
b2FvIJOnts+vXuWW35ofyNbCHzjAzOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCf
w5ICgJ8CuM3vV5ty9bf7KUlOkejzTvv+5PV++elkP9HQ83vqTAws2WwWTxKjEDAO
MAwGA1UdEwEB/wQCMAAwCgYIKoZIzj0EAwIDZwAwZAIwGb0oyM0doP6t3/LSPL5O
DuatEwMYh7WGO+IYTHC8K7EyHBOmCYReKT2+GhV/CLWzAjBNy6UMJTt1tsxJsJqd
MPUIFj+4wZg1AOIb/JoA6M7r33pwLQTrHRxEzVMGfWOkYUw=
-----END CERTIFICATE-----
```


D.1.2. Manufacturer key pair for IDevID signatures

This private key signs IDevID certificates:

-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAgiRoYqKoEcfOfvRvmZ5P5Azn58tuI7nSnIy7OgFnCeiNo+BmbgMho
r6lcU60gwVagBwYFK4EEACKhZANiAATZAH3Rb2FvIJOnts+vXuWW35ofyNbCHzjA
zOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCfw5ICgJ8CuM3vV5ty9bf7KUlOkejz
Tvv+5PV++elkP9HQ83vqTAws2WwWTxI=
-----END EC PRIVATE KEY-----

This public key validates IDevID certificates:

-----BEGIN CERTIFICATE-----
MIIBzzCCAVagAwIBAgIBATAKBggqhkjOPQQDAjBNMRIwEAYKCZImiZPyLGQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3RydW5n
IEhpZ2h3YXkgQ0EwHhcNMTcwMzI2MTYxOTQwWhcNMTkwMzI2MTYxOTQwWjBHMRIw
EAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xFjAU
BgNVBAMMDVVuc3RydW5nIE1BU0EwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAATZAH3R
b2FvIJOnts+vXuWW35ofyNbCHzjAzOi2kWZFE1ByurKImNcNMFGirGnRXIXGqWCf
w5ICgJ8CuM3vV5ty9bf7KUlOkejzTvv+5PV++elkP9HQ83vqTAws2WwWTxKjEDAO
MAwGA1UdEwEB/wQCMAAwCgYIKoZIzj0EAwIDZwAwZAIwGb0oyM0doP6t3/LSPL5O
DuatEwMYh7WGO+IYTHC8K7EyHBOmCYReKT2+GhV/CLWzAjBNy6UMJTt1tsxJsJqd
MPUIFj+4wZg1AOIb/JoA6M7r33pwLQTrHRxEzVMGfWOkYUw=
-----END CERTIFICATE-----

D.1.3. Registrar key pair

The registrar key (or chain) is the representative of the domain owner. This key signs registrar voucher-requests:

-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIF+obiToYYYeMifPsZvrjWJ0yFsCJwIFhpokmT/TULmXoAoGCCqGSM49
AwEHoUQDQgAENWQOzcNMUjP0NrtfeBc0DJLWfeMGgCFdIv6FUz4DifM1ujMBec/g
6W/P6boTmyTGdFOh/8HwKUerL5bpneK8sg==
-----END EC PRIVATE KEY-----

The public key is indicated in a pledge voucher-request to show proximity.

-----BEGIN CERTIFICATE-----
MIIBrjCCATOgAwIBAgIBAzAKBggqhkjOPQQDAzBOMRIwEAYKCZImiZPyLGQBGRYC
Y2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHTAbBgNVBAMMFFVuc3RydW5n
IEZvdW50YWluIENBMB4XDTE3MDkwNTAxMTI0NVoXDTE5MDkwNTAxMTI0NVowQzES
MBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMRIw
EAYDVQQDDAlsb2NhbGhvc3QwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQ1ZA7N
w0xSM/Q2u194FzQMktZ94waAIV0i/oVTPgOJ8zW6MwF5z+Dpb8/puhObJMZ0U6H/
wfApR6svlumd4ryyow0wCzAJBgNVHRMEAjAAMAoGCCqGSM49BAMDA2kAMGYCMQC3
/iTQJ3evYYcgbXhbmzrp64t3QC6qjIeY2jkDx062nuNifVKtyaara3F30AIkKSEC
MQDi29efbTLbdtDk3tecY/rD7V77XaJ6nYCmdDCR54TrSFNLgxvt1lyFM+0fYpYR
c3o=
-----END CERTIFICATE-----

The registrar public certificate as decoded by openssl's x509 utility. Note that the registrar certificate is marked with the cmcRA extension.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
    Signature Algorithm: ecdsa-with-SHA384
        Issuer: DC=ca, DC=sandelman, CN=Unstrung Fountain CA
        Validity
            Not Before: Sep  5 01:12:45 2017 GMT
            Not After : Sep  5 01:12:45 2019 GMT
        Subject: DC=ca, DC=sandelman, CN=localhost
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:35:64:0e:cd:c3:4c:52:33:f4:36:bb:5f:7
8:17:
                    34:0c:92:d6:7d:e3:06:80:21:5d:22:fe:85:5
3:3e:
                    03:89:f3:35:ba:33:01:79:cf:e0:e9:6f:cf:e
9:ba:
                    13:9b:24:c6:74:53:a1:ff:c1:f0:29:47:ab:2
f:96:
                    e9:9d:e2:bc:b2
                ASN1 OID: prime256v1
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
    Signature Algorithm: ecdsa-with-SHA384
         30:66:02:31:00:b7:fe:24:d0:27:77:af:61:87:20:6d:78:
5b:
         9b:3a:e9:eb:8b:77:40:2e:aa:8c:87:98:da:39:03:c7:4e:
b6:
         9e:e3:62:7d:52:ad:c9:a6:ab:6b:71:77:d0:02:24:29:21:
02:
         31:00:e2:db:d7:9f:6d:32:db:76:d0:e4:de:d7:9c:63:fa:
c3:
         ed:5e:fb:5d:a2:7a:9d:80:a6:74:30:91:e7:84:eb:48:53:
4b:
         83:1b:ed:d6:5c:85:33:ed:1f:62:96:11:73:7a
```

D.1.4. Pledge key pair

The pledge has an IDevID key pair built in at manufacturing time:


```
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIBgR6SV+uEvWfl5zCQWZxWjYbMhXPyNqdHJ3KPh11mm4oAoGCCqGSM49
AwEHoUQDQgAEWi/jqPpRJ0JgWghZRgeZlLKutbXVjmnHb+1AYaEF/YQjE2g5FZV8
KjiR/bkEl+l8M4onIC7KHaXKKkuag9S6Tw==
-----END EC PRIVATE KEY-----
```


The public key is used by the registrar to find the MASA. The MASA URL is in an extension described in Section 2.3.


```
-----BEGIN CERTIFICATE-----
MIICBDCCAYugAwIBAgIECe20qTAKBggqhkjOPQQDAjBNMRIwEAYKCZImiZPyLGQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xHDAaBgNVBAMME1Vuc3Ry
dW5nIEhpZ2h3YXkgQ0EwIBcNMTkwNDI0MDIxNjU4WhgPMjk5OTEyMzEwMDAwMDBa
MBwxGjAYBgNVBAUMETAwLWQwLWU1LTAyLTAwLTJkMFkwEwYHKoZIzj0CAQYIKoZI
zj0DAQcDQgAEWi/jqPpRJ0JgWghZRgeZlLKutbXVjmnHb+1AYaEF/YQjE2g5FZV8
KjiR/bkEl+l8M4onIC7KHaXKKkuag9S6T6OBhzCBhDAdBgNVHQ4EFgQUj8KYdUoE
OvJ0kcOIbjEWwgWdDYkwCQYDVR0TBAIwADArBgNVHREEJDAioCAGCSsGAQQBgu5S
AaATDBEwMC1EMC1FNS0wMi0wMC0yRDArBgkrBgEEAYLuUgIEHgwcbWFzYS5ob25l
eWR1a2VzLnNhbmRlbG1hbi5jYTAKBggqhkjOPQQDAgNnADBkAjAmvMjmNgjypDhc
fynMV3kMuIpSKrYzRWr4g3PtTwXDsAe0oitTTj4QtU1bajhOfTkCMGMNbsW2Q41F
z9t6PDVdtOKabBbAP1RVoFTlDQuO9nmLzb5kU+cUqCtPRFZBUXP3kg==
-----END CERTIFICATE-----
```


The pledge public certificate as 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. There is a second Custom Extension is included to provided to contain the EUI48/EUI64 that the pledge will configure as it's layer-2 address (this is non-normative).

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 166573225 (0x9edb4a9)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: DC = ca, DC = sandelman, CN = Unstrung Highway CA
        Validity
            Not Before: Apr 24 02:16:58 2019 GMT
            Not After : Dec 31 00:00:00 2999 GMT
        Subject: serialNumber = 00-d0-e5-02-00-2d
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:5a:2f:e3:a8:fa:51:27:42:60:5a:08:59:46:07:
                    99:94:b2:ae:b5:b5:d5:8e:69:c7:6f:ed:40:61:a1:
                    05:fd:84:23:13:68:39:15:95:7c:2a:38:91:fd:b9:
                    04:97:e9:7c:33:8a:27:20:2e:ca:1d:a5:ca:2a:4b:
                    9a:83:d4:ba:4f
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                8F:C2:98:75:4A:04:3A:F2:74:91:C3:88:6E:31:16:C2:05:9D:0D:89
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Alternative Name:
                othername:<unsupported>
            1.3.6.1.4.1.46930.2:
                ..masa.honeydukes.sandelman.ca
    Signature Algorithm: ecdsa-with-SHA256
         30:64:02:30:26:bc:c8:e6:36:08:f2:a4:38:5c:7f:29:cc:57:
         79:0c:b8:8a:52:2a:b6:33:45:6a:f8:83:73:ed:4f:05:c3:b0:
         07:b4:a2:2b:53:4e:3e:10:b5:4d:5b:6a:38:4e:7d:39:02:30:
         63:0d:6e:c5:b6:43:8d:45:cf:db:7a:3c:35:5d:b4:e2:9a:6c:
         16:c0:3f:54:55:a0:54:e5:0d:0b:8e:f6:79:8b:cd:be:64:53:
         e7:14:a8:2b:4f:44:56:41:51:73:f7:92
```

D.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.

D.2.1. Pledge to Registrar

As described in Section 5.2, the pledge will sign a pledge voucher-request containing the registrar's public key in the proximity-registrar-cert field. The base64 has been wrapped at 60 characters for presentation reasons.


```
-----BEGIN CMS-----
MIIGtQYJKoZIhvcNAQcCoIIGpjCCBqICAQExDTALBglghkgBZQMEAgEwggNRBgkq
hkiG9w0BBwGgggNCBIIDPnsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2hlciI6
eyJhc3NlcnRpb24iOiJwcm94aW1pdHkiLCJjcmVhdGVkLW9uIjoiMjAxOS0wNS0x
NVQxNzoyNTo1NS42NDQtMDQ6MDAiLCJzZXJpYWwtbnVtYmVyIjoiMDAtZDAtZTUt
MDItMDAtMmQiLCJub25jZSI6IlZPVUZULVd3ckV2ME51QVFFSG9WN1EiLCJwcm94
aW1pdHktcmVnaXN0cmFyLWNlcnQiOiJNSUlCMFRDQ0FWYWdBd0lCQWdJQkFqQUtC
Z2dxaGtqT1BRUURBekJ4TVJJd0VBWUtDWkltaVpQeUxHUUJHUllDWTJFeEdUQVhC
Z29Ka2lhSmsvSXNaQUVaRmdsellXNWtaV3h0WVc0eFFEQStCZ05WQkFNTU55TThV
M2x6ZEdWdFZtRnlhV0ZpYkdWNk1IZ3dNREF3TURBd05HWTVNVEZoTUQ0Z1ZXNXpk
SEoxYm1jZ1JtOTFiblJoYVc0Z1EwRXdIaGNOTVRjeE1UQTNNak0wTlRJNFdoY05N
VGt4TVRBM01qTTBOVEk0V2pCRE1SSXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhH
VEFYQmdvSmtpYUprL0lzWkFFWkZnbHpZVzVrWld4dFlXNHhFakFRQmdOVkJBTU1D
V3h2WTJGc2FHOXpkREJaTUJNR0J5cUdTTTQ5QWdFR0NDcUdTTTQ5QXdFSEEwSUFC
SlpsVUhJMHVwL2wzZVpmOXZDQmIrbElub0VNRWdjN1JvK1haQ3RqQUkwQ0QxZkpm
SlIvaEl5eURtSFd5WWlORmJSQ0g5ZnlhcmZremdYNHAwelRpenFqRFRBTE1Ba0dB
MVVkRXdRQ01BQXdDZ1lJS29aSXpqMEVBd01EYVFBd1pnSXhBTFFNTnVyZjh0djUw
bFJPRDVEUVhIRU9KSk5XM1FWMmc5UUVkRFNrMk1ZK0FvU3JCU21HU05qaDRvbEVP
aEV1TGdJeEFKNG5XZk53K0JqYlptS2lJaVVFY1R3SE1oR1ZYYU1IWS9GN24zOXd3
S2NCQlNPbmROUHFDcE9FTGw2YnEzQ1pxUT09In19oIICCDCCAgQwggGLoAMCAQIC
BAnttKkwCgYIKoZIzj0EAwIwTTESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYKCZIm
iZPyLGQBGRYJc2FuZGVsbWFuMRwwGgYDVQQDDBNVbnN0cnVuZyBIaWdod2F5IENB
MCAXDTE5MDQyNDAyMTY1OFoYDzI5OTkxMjMxMDAwMDAwWjAcMRowGAYDVQQFDBEw
MC1kMC1lNS0wMi0wMC0yZDBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABFov46j6
USdCYFoIWUYHmZSyrrW11Y5px2/tQGGhBf2EIxNoORWVfCo4kf25BJfpfDOKJyAu
yh2lyipLmoPUuk+jgYcwgYQwHQYDVR0OBBYEFI/CmHVKBDrydJHDiG4xFsIFnQ2J
MAkGA1UdEwQCMAAwKwYDVR0RBCQwIqAgBgkrBgEEAYLuUgGgEwwRMDAtRDAtRTUt
MDItMDAtMkQwKwYJKwYBBAGC7lICBB4MHG1hc2EuaG9uZXlkdWtlcy5zYW5kZWxt
YW4uY2EwCgYIKoZIzj0EAwIDZwAwZAIwJrzI5jYI8qQ4XH8pzFd5DLiKUiq2M0Vq
+INz7U8Fw7AHtKIrU04+ELVNW2o4Tn05AjBjDW7FtkONRc/bejw1XbTimmwWwD9U
VaBU5Q0LjvZ5i82+ZFPnFKgrT0RWQVFz95IxggErMIIBJwIBATBVME0xEjAQBgoJ
kiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1hbjEcMBoGA1UE
AwwTVW5zdHJ1bmcgSGlnaHdheSBDQQIECe20qTALBglghkgBZQMEAgGgaTAYBgkq
hkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0xOTA1MTUyMTI1
NTVaMC8GCSqGSIb3DQEJBDEiBCAQN2lP7aqwyhmj9qUHt6Qk/SbOTOPXFOwn1wv2
5YGYgDAKBggqhkjOPQQDAgRHMEUCIEYQhHToU0rrhPyQv2fR0TwWePTx2Z1DEhR4
tTl/Dr/ZAiEA47u9+bIz/p6nFJ+wctKHER+ycUzYQF56h9odMo+Ilkc=
-----END CMS-----
```


file: examples/vr_00-D0-E5-02-00-2D.pkcs

The ASN1 decoding of the artifact:

```
    0:d=0  hl=4 l=1717 cons: SEQUENCE
    4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
   15:d=1  hl=4 l=1702 cons: cont [ 0 ]
   19:d=2  hl=4 l=1698 cons: SEQUENCE
   23:d=3  hl=2 l=   1 prim: INTEGER           :01
   26:d=3  hl=2 l=  13 cons: SET
   28:d=4  hl=2 l=  11 cons: SEQUENCE
   30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
   41:d=3  hl=4 l= 849 cons: SEQUENCE
   45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
   56:d=4  hl=4 l= 834 cons: cont [ 0 ]
   60:d=5  hl=4 l= 830 prim: OCTET STRING      :{"ietf-voucher-request:v
  894:d=3  hl=4 l= 520 cons: cont [ 0 ]
  898:d=4  hl=4 l= 516 cons: SEQUENCE
  902:d=5  hl=4 l= 395 cons: SEQUENCE
  906:d=6  hl=2 l=   3 cons: cont [ 0 ]
  908:d=7  hl=2 l=   1 prim: INTEGER           :02
  911:d=6  hl=2 l=   4 prim: INTEGER           :09EDB4A9
  917:d=6  hl=2 l=  10 cons: SEQUENCE
  919:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
  929:d=6  hl=2 l=  77 cons: SEQUENCE
  931:d=7  hl=2 l=  18 cons: SET
  933:d=8  hl=2 l=  16 cons: SEQUENCE
  935:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  947:d=9  hl=2 l=   2 prim: IA5STRING         :ca
  951:d=7  hl=2 l=  25 cons: SET
  953:d=8  hl=2 l=  23 cons: SEQUENCE
  955:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  967:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
  978:d=7  hl=2 l=  28 cons: SET
  980:d=8  hl=2 l=  26 cons: SEQUENCE
  982:d=9  hl=2 l=   3 prim: OBJECT            :commonName
  987:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
 1008:d=6  hl=2 l=  32 cons: SEQUENCE
 1010:d=7  hl=2 l=  13 prim: UTCTIME           :190424021658Z
 1025:d=7  hl=2 l=  15 prim: GENERALIZEDTIME   :29991231000000Z
 1042:d=6  hl=2 l=  28 cons: SEQUENCE
 1044:d=7  hl=2 l=  26 cons: SET
 1046:d=8  hl=2 l=  24 cons: SEQUENCE
 1048:d=9  hl=2 l=   3 prim: OBJECT            :serialNumber
 1053:d=9  hl=2 l=  17 prim: UTF8STRING        :00-d0-e5-02-00-2d
 1072:d=6  hl=2 l=  89 cons: SEQUENCE
 1074:d=7  hl=2 l=  19 cons: SEQUENCE
 1076:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 1085:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
 1095:d=7  hl=2 l=  66 prim: BIT STRING
```

```
 1163:d=6  hl=3 l= 135 cons: cont [ 3 ]
 1166:d=7  hl=3 l= 132 cons: SEQUENCE
 1169:d=8  hl=2 l=  29 cons: SEQUENCE
 1171:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Ident
 1176:d=9  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:04148FC298754A
 1200:d=8  hl=2 l=   9 cons: SEQUENCE
 1202:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 1207:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
 1211:d=8  hl=2 l=  43 cons: SEQUENCE
 1213:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Alternati
 1218:d=9  hl=2 l=  36 prim: OCTET STRING      [HEX DUMP]:3022A02006092B
 1256:d=8  hl=2 l=  43 cons: SEQUENCE
 1258:d=9  hl=2 l=   9 prim: OBJECT            :1.3.6.1.4.1.46930.2
 1269:d=9  hl=2 l=  30 prim: OCTET STRING      [HEX DUMP]:0C1C6D6173612E
 1301:d=5  hl=2 l=  10 cons: SEQUENCE
 1303:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1313:d=5  hl=2 l= 103 prim: BIT STRING
 1418:d=3  hl=4 l= 299 cons: SET
 1422:d=4  hl=4 l= 295 cons: SEQUENCE
 1426:d=5  hl=2 l=   1 prim: INTEGER           :01
 1429:d=5  hl=2 l=  85 cons: SEQUENCE
 1431:d=6  hl=2 l=  77 cons: SEQUENCE
 1433:d=7  hl=2 l=  18 cons: SET
 1435:d=8  hl=2 l=  16 cons: SEQUENCE
 1437:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1449:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 1453:d=7  hl=2 l=  25 cons: SET
 1455:d=8  hl=2 l=  23 cons: SEQUENCE
 1457:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1469:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 1480:d=7  hl=2 l=  28 cons: SET
 1482:d=8  hl=2 l=  26 cons: SEQUENCE
 1484:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 1489:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
 1510:d=6  hl=2 l=   4 prim: INTEGER           :09EDB4A9
 1516:d=5  hl=2 l=  11 cons: SEQUENCE
 1518:d=6  hl=2 l=   9 prim: OBJECT            :sha256
 1529:d=5  hl=2 l= 105 cons: cont [ 0 ]
 1531:d=6  hl=2 l=  24 cons: SEQUENCE
 1533:d=7  hl=2 l=   9 prim: OBJECT            :contentType
 1544:d=7  hl=2 l=  11 cons: SET
 1546:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 1557:d=6  hl=2 l=  28 cons: SEQUENCE
 1559:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
 1570:d=7  hl=2 l=  15 cons: SET
 1572:d=8  hl=2 l=  13 prim: UTCTIME           :190515212555Z
 1587:d=6  hl=2 l=  47 cons: SEQUENCE
 1589:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
```

```
 1600:d=7  hl=2 l=  34 cons: SET
 1602:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:1037694FEDAAB0
 1636:d=5  hl=2 l=  10 cons: SEQUENCE
 1638:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1648:d=5  hl=2 l=  71 prim: OCTET STRING      [HEX DUMP]:30450220461084
```

The JSON contained in the voucher request:


```
{"ietf-voucher-request:voucher":{"assertion":"proximity","cr
eated-on":"2019-05-15T17:25:55.644-04:00","serial-number":"0
0-d0-e5-02-00-2d","nonce":"VOUFT-WwrEv0NuAQEHoV7Q","proximit
y-registrar-cert":"MIIB0TCCAVagAwIBAgIBAjAKBggqhkjOPQQDAzBxM
RIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtY
W4xQDA+BgNVBAMMNyM8U3lzdGVtVmFyaWFibGU6MHgwMDAwMDAwNGY5MTFhM
D4gVW5zdHJ1bmcgRm91bnRhaW4gQ0EwHhcNMTcxMTA3MjM0NTI4WhcNMTkxM
TA3MjM0NTI4WjBDMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZ
AEZFglzYW5kZWxtYW4xEjAQBgNVBAMMCWxvY2FsaG9zdDBZMBMGByqGSM49A
gEGCCqGSM49AwEHA0IABJZlUHI0up/l3eZf9vCBb+lInoEMEgc7Ro+XZCtjA
I0CD1fJfJR/hIyyDmHWyYiNFbRCH9fyarfkzgX4p0zTizqjDTALMAkGA1UdE
wQCMAAwCgYIKoZIzj0EAwMDaQAwZgIxALQMNurf8tv50lROD5DQXHEOJJNW3
QV2g9QEdDSk2MY+AoSrBSmGSNjh4olEOhEuLgIxAJ4nWfNw+BjbZmKiIiUEc
TwHMhGVXaMHY/F7n39wwKcBBSOndNPqCpOELl6bq3CZqQ=="}}
```


D.2.2. Registrar to MASA

As described in Section 5.5 the registrar will sign a registrar voucher-request, and will include pledge’s voucher request in the prior-signed-voucher-request.


```
-----BEGIN CMS-----
MIIPkwYJKoZIhvcNAQcCoIIPhDCCD4ACAQExDTALBglghkgBZQMEAgEwggnUBgkq
hkiG9w0BBwGgggnFBIIJwXsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2hlciI6
eyJhc3NlcnRpb24iOiJwcm94aW1pdHkiLCJjcmVhdGVkLW9uIjoiMjAxOS0wNS0x
NVQyMToyNTo1NS43NThaIiwic2VyaWFsLW51bWJlciI6IjAwLWQwLWU1LTAyLTAw
LTJkIiwibm9uY2UiOiJWT1VGVC1Xd3JFdjBOdUFRRUhvVjdRIiwicHJpb3Itc2ln
bmVkLXZvdWNoZXItcmVxdWVzdCI6Ik1JSUd0UVlKS29aSWh2Y05BUWNDb0lJR3Bq
Q0NCcUlDQVFFeERUQUxCZ2xnaGtnQlpRTUVBZ0V3Z2dOUkJna3Foa2lHOXcwQkJ3
R2dnZ05DQklJRFBuc2lhV1YwWmkxMmIzVmphR1Z5TFhKbGNYVmxjM1E2ZG05MVky
aGxjaUk2ZXlKaGMzTmxjblJwYjI0aU9pSndjbTk0YVcxcGRIa2lMQ0pqY21WaGRH
VmtMVzl1SWpvaU1qQXhPUzB3TlMweE5WUXhOem95TlRvMU5TNDJORFF0TURRNk1E
QWlMQ0p6WlhKcFlXd3RiblZ0WW1WeUlqb2lNREF0WkRBdFpUVXRNREl0TURBdE1t
UWlMQ0p1YjI1alpTSTZJbFpQVlVaVUxWZDNja1YyTUU1MVFWRkZTRzlXTjFFaUxD
SndjbTk0YVcxcGRIa3RjbVZuYVhOMGNtRnlMV05sY25RaU9pSk5TVWxDTUZSRFEw
RldZV2RCZDBsQ1FXZEpRa0ZxUVV0Q1oyZHhhR3RxVDFCUlVVUkJla0o0VFZKSmQw
VkJXVXRDV21sMVNGQlpiRXhIVVVKSFVWbERXVEpGZUVkVVFWaENaMjlLYTJsaFNt
c3ZTWE5hUVVWWVJtZHNlbGxYTld0YVYzaDBXVmMwZUZGRVFTdENaMDVXUWtGTlRV
NTVUVGhWTTJ4NlpFZFdkRlp0UmxsaFYwWnBZa2RWTmsxSVozZE5SRUYzVFVSQmQw
NUhXVFZOVkVab1RVUTBaMVpYTlhwa1NFb3hZbTFqWjFKdE9URmliRnBvYVc0Z1Ewox
```

```
RXdSWGRJYUdOT1RWUmplRTFVUVROTmFrMHdUbFJKTkZkb1kwNU5WR3Q0VFZSQk0w
MXFUVEJPVkVrMFYycENSRTFTU1hkRlFWbExRMXBKYldsYVVIbE1SMUZDUjFKWlEx
a3lSWGhIVkVGWVFtZHZTbXRwWVVwckwwbHpXa0ZGV2tabmJIcFpWelZyV2xkNGRG
bFhOSGhGYWtGUlFtZE9Wa0pCVFUxRFYzaDJXVEpHYzJGSE9YcGtSRUphVFVKTlIw
SjVjVWRUVFRRNVFXZEZSME5EY1VkVFRUUTVRWGRGU0VFd1NVRkNTbHBzVlVoSk1I
VndMMnd6WlZwbU9YWkRRbUlyYkVsdWIwVk5SV2RqTjFKdksxaGFRM1JxUVVrd1EW
UXhaa3BtU2xJdmFFbDVVVJ0U0ZkNVdXbE9SbUpTUTBnNVpubGhjbVpyZW1kWU5I
QXdlbFJwZW5GcVJGUkJURFCYTBkQk1WVmtSWGRSUTAxQlFYZERaMWxKUzI5YVNY
cHFNRVZCZDAxRVlWRkJkMXBuU1hoQlRGRk5kblZ5WmpoMGRqVXdiRkpQUkRWVVW
aElSVTlLU2s1WE0xRldNbWM1VVVWa1JGTnJNazFaS3pCdlUzSkNVMjFIVTA1cWFE
UnZiRVZQYUVWMVRHZEplRUZLTkc1WFpreE9NVXBMTkpxWWxdFMybEphVlZGWTFNF
MW9SMVpZWVUxSVdTOUdOMjR6T1hkM1MyTkNRbE5QYm1ST1VIRkRjRTlGVEd3Mllu
RXpRMXB4VVQwOUluMTlvSUlDQ0RDQ0FnUXdnZ0dMb0FNQ0FRSUNCQW50dEtrd0Nn
WUlLb1pJemowRUF3SXdUVEVTTUJBR0NnbVNKb21UOGl4a0FSa1dBbU5vTVJrd0Z3
WUtDWkltaVpQeUxHUUJHUllKYzJGdVpHVnNiV0Z1TVJ3d0dnWURWUVFERUJOVmJu
TjBjblZ1WnlCSWFXZG9kMkY1SUVOQk1DQVhEVEU1TURReU5EQXlNVFkxT0ZvWUR6
STVPVGt4TWpNeE1EQXdNREF3V2pCY01TRXdId1lEVlFRRkRCRXdNQzFEUXpBekxURUMxbE5T
MHdNaTB3TUMweVpEQlZOQk1HQ25sxR1NNNDlBZ0VHQ0NxR1NNNDlBd0VIQTBJQUJKQUJG
b3Y0Nmo2VVNkQ1lGb0lXVVlIbVpTeXJyVzExWTVweDIvdFFHR2hCZjjJFSXhOb09S
V1ZmQ280a2YyNUJKZnBmRE9LSnlBdXloMmx5aXBMbW9QVXVrK2pnWWN3Z1lRd0hR
WURWUjBPQkJZRUZJL0NtSFZLQkRyeWRKSERpRzR4RnNJRm5RMkpNQWtHQTFVZEV3
UUNNQUF3S3dZRFZSMFJCQ1F3SXFBZ0Jna3JCZ0VFQVlMdVVnR2dFd1lSaE1NVVhkFJE
QXRSVFV0TURJdE1EQXRNa1F3S3dZRFZSkt3WUJCQUdDN2xJQ0NE1IRzFoYzJFdWFH
OXVaWGxrZFd0bGN5NXpZVzVrVlw4dFlXNHVZMkV3Q2dZSUtvWkl6ajBFQXdJRFp3
QXdaQUl3SnJ6STVqWUk4cVE0WEg4cHpGZDVETGlLVWlxMk0wVnErSU56N1U4Rnc3
QUh0S0lyVTA0K0VMVk5XMm80VG4wNUFqQmpEVzdGdGtPTlJjL2JlancxWGJUaW1t
d1d3RDlVVmFCVTVRMExxdlo1aTgyK1pGUG5GS2dyVDBSV1FWRno5NUl4Z2dFck1J
SUJKd0lCQVRCVk1FMHhFakFRQmdvSmtpYUprL0lzWkFFWkZnSmpZVEVaTUJjR0Nn
bVNKb21UOGl4a0FSa1dDWE5oYm1SbGJHMWhiakVjTUJvR0ExVUVBd3dUVlc1emRI
SjFibWNnU0dsbmFIZGhlU0JEUVFJRUNlMjBxVEFMQmdsZ2hrZ0JaUU1FQWdHZ2FU
QVlCZ2txaGtpRzl3MEJDUU14Q3dZSktvWklodmNOQVFjQk1Cd0dDU3FHU0liM0RR
RUpCVEVQRncweE9UQTFNVFV5TVRNNQzhHQ1NxR1NJYjNEUUVKQkRFaUJDRFFUTkJD
QVFOMmxQN2Fxd3lobWo5cVVIdDZRay9TYk9UT1BYRk93bjF3bjF3WUdZZ0RBS0Jn
Z3Foa2pPUFFREFnUkhNRVVDSVFBZWhIVG9VMHJyaFB5UXYyZlIwVHdXZVBDdDJa
MURFaFI0dFRsL0RyL1pBaUVBNDd1OStiSXovcDZuRkord2N2N0RTS0hFUit5Y1V6WVFG
NTZoOW9kTW8rSWxrYz0ifX2gggRCMIIB0TCCAVagAwIBAgIBAjAKBggqhkjOPQQD
AzBxMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxt
YW4xQDA+BgNVBAMMNyM8U3lzdGVtVmFyaWFibGU6MHgwMDAwMDAwNGY5MTFhMD4g
VW5zdHJ1bmcgRm91bnRhaW4gQ0EwHhcNMTcxMTA3MjM0NTI4WhcNMTkxMTA3MjM0
NTI4WjBDMRIwEAYKCZImiZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5k
ZWxtYW4xEjAQBgNVBAMMCWxvY2FsaG9zdDBZMBMGByqGSM49AgEGCCqGSM49AwEH
A0IABJZlUHI0up/l3eZf9vCBb+lInoEMEgc7Ro+XZCtjAI0CD1fJfJR/hIyyDmHW
yYiNFbRCH9fyarfkzgX4p0zTizqjDTALMAkGA1UdEwQCMAAwCgYIKoZIzj0EAwMD
aQAwZgIxALQMNurf8tv50lROD5DQXHEOJJNW3QV2g9QEdDSk2MY+AoSrBSmGSNjh
4olEOhEuLgIxAJ4nWfNw+BjbZmKiIiUEcTwHMhGVXaMHY/F7n39wwKcBBSOndNPq
CpOELl6bq3CZqTCCAmkwggHvoAMCAQICAQMwCgYIKoZIzj0EAwIwbTESMBAGCgmS
JomT8ixkARkWAmNhMRkwFwYKCZImiZPyLGQBGRYJc2FuZGVsbWFuMTwwOgYDVQQD
DDNmb3VudGFpbi10ZXN0LmV4YW1wbGUuY29tIFVuc3RydW5nIEZvdW50YWluIFJv
```

```
b3QgQ0EwHhcNMTkwMTEzMjI1NDQ0WhcNMjEwMTEyMjI1NDQ0WjBtMRIwEAYKCZIm
iZPyLGQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMM
M2ZvdW50YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9v
dCBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABBt/WboXwxq8Zo2MbODD+jFxD2X2
IpG9t1aAB9vfuHqlRU15ikaXGVmWMbGPaX0yvjzIPltjtUb2qNVvm/nA89O5FD9y
R1Gkdt3S8L/1yo8wAX/4wl/T9SADRIuL8gdstKNjMGEwDwYDVR0TAQH/BAUwAwEB
/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFLml9ssR4QekSSynCMZ8ELyHs3Qm
MB8GA1UdIwQYMBaAFLml9ssR4QekSSynCMZ8ELyHs3QmMAoGCCqGSM49BAMCA2gA
MGUCMAviLdbfd6AZdsOxNgf7D15WFmGC1JkHeEbT/0w4UXz6q/48S71/IMbSXRWH
aNxiJwIxAOCRjtlN+VSmCLTvWwMTxnSpIuqMr/O1y2Z8rl459VRFphWPdbf4i0qE
cwu0u4JzpDGCAUwwggFIAgEBMHYwcTESMBAGCgmSJomT8ixkARkWAmNhMRkwFwYK
CZImiZPyLGQBGRYJc2FuZGVsbWFuMUAwPgYDVQQDDDcjPFN5c3RlbVZhcmlhYmxl
OjB4MDAwMDAwMDRmOTExYTA+IFVuc3RydW5nIEZvdW50YWluIENBAgECMAsGCWCG
SAFlAwQCAaBpMBgGCSqGSIb3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkF
MQ8XDTE5MDUxNTIxMjU1NVowLwYJKoZIhvcNAQkEMSIEIFBQjMmWzZOEkRHXrVAS
snJwgQ26goyvOAtUFYs3MstMMAoGCCqGSM49BAMCBEcwRQIgBthbhEmgbqZbYDkD
zxHXLzJ5eusWplzHKqZyxNpzaR8CIQC3UtMu0QsXoUpYL016iTsbd7Eedi8IfnwQ
akExfhh0ew==
-----END CMS-----
```


file: examples/parboiled_vr_00_D0-E5-02-00-2D.pkcs

The ASN1 decoding of the artifact:

```
  0:d=0  hl=4 l=3987 cons: SEQUENCE
  4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
 15:d=1  hl=4 l=3972 cons: cont [ 0 ]
 19:d=2  hl=4 l=3968 cons: SEQUENCE
 23:d=3  hl=2 l=   1 prim: INTEGER           :01
 26:d=3  hl=2 l=  13 cons: SET
 28:d=4  hl=2 l=  11 cons: SEQUENCE
 30:d=5  hl=2 l=   9 prim: OBJECT            :sha256
 41:d=3  hl=4 l=2516 cons: SEQUENCE
 45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 56:d=4  hl=4 l=2501 cons: cont [ 0 ]
 60:d=5  hl=4 l=2497 prim: OCTET STRING      :{"ietf-voucher-request:v
2561:d=3  hl=4 l=1090 cons: cont [ 0 ]
2565:d=4  hl=4 l= 465 cons: SEQUENCE
2569:d=5  hl=4 l= 342 cons: SEQUENCE
2573:d=6  hl=2 l=   3 cons: cont [ 0 ]
2575:d=7  hl=2 l=   1 prim: INTEGER           :02
2578:d=6  hl=2 l=   1 prim: INTEGER           :02
2581:d=6  hl=2 l=  10 cons: SEQUENCE
2583:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA384
2593:d=6  hl=2 l= 113 cons: SEQUENCE
2595:d=7  hl=2 l=  18 cons: SET
2597:d=8  hl=2 l=  16 cons: SEQUENCE
2599:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
```

```
 2611:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 2615:d=7  hl=2 l=  25 cons: SET
 2617:d=8  hl=2 l=  23 cons: SEQUENCE
 2619:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 2631:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 2642:d=7  hl=2 l=  64 cons: SET
 2644:d=8  hl=2 l=  62 cons: SEQUENCE
 2646:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 2651:d=9  hl=2 l=  55 prim: UTF8STRING        :#<SystemVariable:0x00000
 2708:d=6  hl=2 l=  30 cons: SEQUENCE
 2710:d=7  hl=2 l=  13 prim: UTCTIME           :171107234528Z
 2725:d=7  hl=2 l=  13 prim: UTCTIME           :191107234528Z
 2740:d=6  hl=2 l=  67 cons: SEQUENCE
 2742:d=7  hl=2 l=  18 cons: SET
 2744:d=8  hl=2 l=  16 cons: SEQUENCE
 2746:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 2758:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 2762:d=7  hl=2 l=  25 cons: SET
 2764:d=8  hl=2 l=  23 cons: SEQUENCE
 2766:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 2778:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 2789:d=7  hl=2 l=  18 cons: SET
 2791:d=8  hl=2 l=  16 cons: SEQUENCE
 2793:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 2798:d=9  hl=2 l=   9 prim: UTF8STRING        :localhost
 2809:d=6  hl=2 l=  89 cons: SEQUENCE
 2811:d=7  hl=2 l=  19 cons: SEQUENCE
 2813:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 2822:d=8  hl=2 l=   8 prim: OBJECT            :prime256v1
 2832:d=7  hl=2 l=  66 prim: BIT STRING
 2900:d=6  hl=2 l=  13 cons: cont [ 3 ]
 2902:d=7  hl=2 l=  11 cons: SEQUENCE
 2904:d=8  hl=2 l=   9 cons: SEQUENCE
 2906:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 2911:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
 2915:d=5  hl=2 l=  10 cons: SEQUENCE
 2917:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA384
 2927:d=5  hl=2 l= 105 prim: BIT STRING
 3034:d=4  hl=4 l= 617 cons: SEQUENCE
 3038:d=5  hl=4 l= 495 cons: SEQUENCE
 3042:d=6  hl=2 l=   3 cons: cont [ 0 ]
 3044:d=7  hl=2 l=   1 prim: INTEGER           :02
 3047:d=6  hl=2 l=   1 prim: INTEGER           :03
 3050:d=6  hl=2 l=  10 cons: SEQUENCE
 3052:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 3062:d=6  hl=2 l= 109 cons: SEQUENCE
 3064:d=7  hl=2 l=  18 cons: SET
 3066:d=8  hl=2 l=  16 cons: SEQUENCE
```

```
 3068:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3080:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 3084:d=7  hl=2 l=  25 cons: SET
 3086:d=8  hl=2 l=  23 cons: SEQUENCE
 3088:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3100:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 3111:d=7  hl=2 l=  60 cons: SET
 3113:d=8  hl=2 l=  58 cons: SEQUENCE
 3115:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 3120:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
 3173:d=6  hl=2 l=  30 cons: SEQUENCE
 3175:d=7  hl=2 l=  13 prim: UTCTIME           :190113225444Z
 3190:d=7  hl=2 l=  13 prim: UTCTIME           :210112225444Z
 3205:d=6  hl=2 l= 109 cons: SEQUENCE
 3207:d=7  hl=2 l=  18 cons: SET
 3209:d=8  hl=2 l=  16 cons: SEQUENCE
 3211:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3223:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 3227:d=7  hl=2 l=  25 cons: SET
 3229:d=8  hl=2 l=  23 cons: SEQUENCE
 3231:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3243:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 3254:d=7  hl=2 l=  60 cons: SET
 3256:d=8  hl=2 l=  58 cons: SEQUENCE
 3258:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 3263:d=9  hl=2 l=  51 prim: UTF8STRING        :fountain-test.example.co
 3316:d=6  hl=2 l= 118 cons: SEQUENCE
 3318:d=7  hl=2 l=  16 cons: SEQUENCE
 3320:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 3329:d=8  hl=2 l=   5 prim: OBJECT            :secp384r1
 3336:d=7  hl=2 l=  98 prim: BIT STRING
 3436:d=6  hl=2 l=  99 cons: cont [ 3 ]
 3438:d=7  hl=2 l=  97 cons: SEQUENCE
 3440:d=8  hl=2 l=  15 cons: SEQUENCE
 3442:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 3447:d=9  hl=2 l=   1 prim: BOOLEAN           :255
 3450:d=9  hl=2 l=   5 prim: OCTET STRING      [HEX DUMP]:30030101FF
 3457:d=8  hl=2 l=  14 cons: SEQUENCE
 3459:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Key Usage
 3464:d=9  hl=2 l=   1 prim: BOOLEAN           :255
 3467:d=9  hl=2 l=   4 prim: OCTET STRING      [HEX DUMP]:03020106
 3473:d=8  hl=2 l=  29 cons: SEQUENCE
 3475:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Subject Key Ident
 3480:d=9  hl=2 l=  22 prim: OCTET STRING      [HEX DUMP]:0414B9A5F6CB11
 3504:d=8  hl=2 l=  31 cons: SEQUENCE
 3506:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Authority Key Ide
 3511:d=9  hl=2 l=  24 prim: OCTET STRING      [HEX DUMP]:30168014B9A5F6
 3537:d=5  hl=2 l=  10 cons: SEQUENCE
```

```
 3539:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 3549:d=5  hl=2 l= 104 prim: BIT STRING
 3655:d=3  hl=4 l= 332 cons: SET
 3659:d=4  hl=4 l= 328 cons: SEQUENCE
 3663:d=5  hl=2 l=   1 prim: INTEGER           :01
 3666:d=5  hl=2 l= 118 cons: SEQUENCE
 3668:d=6  hl=2 l= 113 cons: SEQUENCE
 3670:d=7  hl=2 l=  18 cons: SET
 3672:d=8  hl=2 l=  16 cons: SEQUENCE
 3674:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3686:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 3690:d=7  hl=2 l=  25 cons: SET
 3692:d=8  hl=2 l=  23 cons: SEQUENCE
 3694:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 3706:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 3717:d=7  hl=2 l=  64 cons: SET
 3719:d=8  hl=2 l=  62 cons: SEQUENCE
 3721:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 3726:d=9  hl=2 l=  55 prim: UTF8STRING        :#<SystemVariable:0x00000
 3783:d=6  hl=2 l=   1 prim: INTEGER           :02
 3786:d=5  hl=2 l=  11 cons: SEQUENCE
 3788:d=6  hl=2 l=   9 prim: OBJECT            :sha256
 3799:d=5  hl=2 l= 105 cons: cont [ 0 ]
 3801:d=6  hl=2 l=  24 cons: SEQUENCE
 3803:d=7  hl=2 l=   9 prim: OBJECT            :contentType
 3814:d=7  hl=2 l=  11 cons: SET
 3816:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 3827:d=6  hl=2 l=  28 cons: SEQUENCE
 3829:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
 3840:d=7  hl=2 l=  15 cons: SET
 3842:d=8  hl=2 l=  13 prim: UTCTIME           :190515212555Z
 3857:d=6  hl=2 l=  47 cons: SEQUENCE
 3859:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
 3870:d=7  hl=2 l=  34 cons: SET
 3872:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:50508CC996CD93
 3906:d=5  hl=2 l=  10 cons: SEQUENCE
 3908:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 3918:d=5  hl=2 l=  71 prim: OCTET STRING      [HEX DUMP]:3045022006D85B
```

D.2.3. MASA to Registrar

The MASA will return a voucher to the registrar, to be relayed to the pledge.

```
-----BEGIN CMS-----
MIIGsgYJKoZIhvcNAQcCoIIGozCCBp8CAQExDTALBglghkgBZQMEAgEwggNABgkq
hkiG9w0BBwGgggMxBIIDLXsiaWV0Zi12b3VjaGVyOnZvdWNoZXIiOnsiYXNzZXJ0
aW9uIjoibG9nZ2VkIiwiY3JlYXRlZC1vbiI6IjIwMTktMDUtMTZUMDI6NTE6NDIu
Njk3KzAwOjAwIiwic2VyaWFsLW51bWJlciI6IjAwLWQwLWU1LTAyLTAwLTJkIiwi
bm9uY2UiOiJHWmUtT2pvZXJwS0VNNFNNN1N6UzlnIiwicGlubmVkLWRvbWFpbi1j
ZXJ0IjoiTUlJQjBUQ0NBVmFnQXdJQkFnSUJBakFLQmdncWhrak9QUVFEQXpCeE1S
SXdFQVlLQ1pJbWlaUHlMR1FCR1JZQ1kyRXhHVEFYQmdvSmtpYUprL0lzWkFFWkZn
bHpZVzVrWld4dFlXNHhRREErQmdOVkJBTU1OeU04VTNsemRHVnRWbUZ5YVdGaWJH
VTZNSGd3TURBd01EQXdOR1k1TVRGaE1ENGdWVzV6ZEhKMWJtY2dSbTkxYm5SaGFX
NGdRMEV3SGhjTk1UY3hNVEEzTWpNME5USTRXaGNOTVRreE1UQTNNak0wTlRJNFdq
QkRNUkl3RUFZS0NaSW1pWlB5TEdRQkdSWUNZMkV4R1RBWEJnb0praWFKay9Jc1pB
RVpGZ2x6WVc1a1pXeHRZVzR4RWpBUUJnTlZCQU1NQ1d4dlkyRnNhRzl6ZERCWk1C
TUdCeXFHU000OUFnRUdDQ3FHU000OUF3RUhBMElBQkpabFVISTB1cC9sM2VaZjl2
Q0JiK2xJbm9FTUVnYzdSbytYWkN0akFJMENEMWZKZkpSL2hJeXlEbUhXeVlpTkZi
UkNIOWZ5YXJma3pnWDRwMHpUaXpxakRUQUxNQWtHQTFVZEV3UUNNQUF3Q2dZSUtv
Wkl6ajBFQXdNRGFRQXdaZ0l4QUxRTU51cmY4dHY1MGxST0Q1RFFYSEVPSkpOVzNR
VjJnOVFFZERTazJNWStBb1NyQlNtR1NOamg0b2xFT2hFdUxnSXhBSjRuV2ZOdytC
amJabUtpSWlVRWNUd0hNaEdWWGFNSFkvRjduMzl3d0tjQkJTT25kTlBxQ3BPRUxs
NmJxM0NacVE9PSJ9faCCAfUwggHxMIIBeKADAgECAgQjzIkTMAoGCCqGSM49BAMC
ME0xEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRlbG1h
bjEcMBoGA1UEAwwTVW5zdHJ1bmcgSGlnaHdheSBDQTAeFw0xOTA0MjMyMzIxMDda
Fw0xOTA1MjQwOTIxMDdaMGYxDzANBgNVBAYTBkNhbmFkYTESMBAGA1UECgwJU2Fu
ZGVsbWFuMRMwEQYDVQQLDApob25leWR1a2VzMSowKAYDVQQDDCFtYXNhLmhvbmV5
ZHVrZXMuc2FuZGVsbWFuLmNhIE1BU0EwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAQ1
/2UdVp8zVmgADoBNql7LcPlJsEaaVAogYEqABikNOkoTO3oPjIQfNBxtGfRFzBXx
gihzkTH58r8SW1L/Mej8AFqhB4SZyyjmWURdzD71Ju0M+tRritWf7T+QGaE+fcWj
EDAOMAwGA1UdEwEB/wQCMAAwCgYIKoZIzj0EAwIDZwAwZAIwOMlNOMNYEZo4yLW4
iRltDL8uirmjMdtVmmVYzqYHSindjP0a3pXQkQZ5LLARoSRWAjBTxsnv6ya5HpZI
IWcspDPZGlOSDPm7nuRJSDkgWqevxLI4+9nmIhsfMBsDvz1DJhAxggFMMIIBSAIB
ATBVME0xEjAQBgoJkiaJk/IsZAEZFgJjYTEZMBcGCgmSJomT8ixkARkWCXNhbmRl
bG1hbjEcMBoGA1UEAwwTVW5zdHJ1bmcgSGlnaHdheSBDQQIEI8yJEzALBglghkgB
ZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEP
Fw0xOTA1MTYwMjUxNDJaMC8GCSqGSIb3DQEJBDEiBCCYRh4i21QjEjEk8leRLSVA
x/EVY5g1bM40QM21oR4c2DAKBggqhkjOPQQDAgRoMGYCMQCYYOiSbIlED4nAN0iL
e4S8ixWAZ9SXpGv77bB/G4fTTVTN35mnAeYBfeNfhC6/kOECMQDqlkCmwQJQDdEL
asj1ISinJ/FnZjjgOMz9MXOmGNGIfw9v2VBb9mVyhsOSMcqlVig=
-----END CMS-----
```


file: examples/voucher_00-D0-E5-02-00-2D.pkcs

The ASN1 decoding of the artifact:

```
 0:d=0  hl=4 l=1714 cons: SEQUENCE
 4:d=1  hl=2 l=   9 prim: OBJECT            :pkcs7-signedData
15:d=1  hl=4 l=1699 cons: cont [ 0 ]
19:d=2  hl=4 l=1695 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= 832 cons: SEQUENCE
   45:d=4  hl=2 l=   9 prim: OBJECT            :pkcs7-data
   56:d=4  hl=4 l= 817 cons: cont [ 0 ]
   60:d=5  hl=4 l= 813 prim: OCTET STRING      :{"ietf-voucher:voucher":
  877:d=3  hl=4 l= 501 cons: cont [ 0 ]
  881:d=4  hl=4 l= 497 cons: SEQUENCE
  885:d=5  hl=4 l= 376 cons: SEQUENCE
  889:d=6  hl=2 l=   3 cons: cont [ 0 ]
  891:d=7  hl=2 l=   1 prim: INTEGER           :02
  894:d=6  hl=2 l=   4 prim: INTEGER           :23CC8913
  900:d=6  hl=2 l=  10 cons: SEQUENCE
  902:d=7  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
  912:d=6  hl=2 l=  77 cons: SEQUENCE
  914:d=7  hl=2 l=  18 cons: SET
  916:d=8  hl=2 l=  16 cons: SEQUENCE
  918:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  930:d=9  hl=2 l=   2 prim: IA5STRING         :ca
  934:d=7  hl=2 l=  25 cons: SET
  936:d=8  hl=2 l=  23 cons: SEQUENCE
  938:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
  950:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
  961:d=7  hl=2 l=  28 cons: SET
  963:d=8  hl=2 l=  26 cons: SEQUENCE
  965:d=9  hl=2 l=   3 prim: OBJECT            :commonName
  970:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
  991:d=6  hl=2 l=  30 cons: SEQUENCE
  993:d=7  hl=2 l=  13 prim: UTCTIME           :190423232107Z
 1008:d=7  hl=2 l=  13 prim: UTCTIME           :190524092107Z
 1023:d=6  hl=2 l= 102 cons: SEQUENCE
 1025:d=7  hl=2 l=  15 cons: SET
 1027:d=8  hl=2 l=  13 cons: SEQUENCE
 1029:d=9  hl=2 l=   3 prim: OBJECT            :countryName
 1034:d=9  hl=2 l=   6 prim: PRINTABLESTRING   :Canada
 1042:d=7  hl=2 l=  18 cons: SET
 1044:d=8  hl=2 l=  16 cons: SEQUENCE
 1046:d=9  hl=2 l=   3 prim: OBJECT            :organizationName
 1051:d=9  hl=2 l=   9 prim: UTF8STRING        :Sandelman
 1062:d=7  hl=2 l=  19 cons: SET
 1064:d=8  hl=2 l=  17 cons: SEQUENCE
 1066:d=9  hl=2 l=   3 prim: OBJECT            :organizationalUnitName
 1071:d=9  hl=2 l=  10 prim: UTF8STRING        :honeydukes
 1083:d=7  hl=2 l=  42 cons: SET
 1085:d=8  hl=2 l=  40 cons: SEQUENCE
 1087:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 1092:d=9  hl=2 l=  33 prim: UTF8STRING        :masa.honeydukes.sandelma
```

```
 1127:d=6  hl=2 l= 118 cons: SEQUENCE
 1129:d=7  hl=2 l=  16 cons: SEQUENCE
 1131:d=8  hl=2 l=   7 prim: OBJECT            :id-ecPublicKey
 1140:d=8  hl=2 l=   5 prim: OBJECT            :secp384r1
 1147:d=7  hl=2 l=  98 prim: BIT STRING
 1247:d=6  hl=2 l=  16 cons: cont [ 3 ]
 1249:d=7  hl=2 l=  14 cons: SEQUENCE
 1251:d=8  hl=2 l=  12 cons: SEQUENCE
 1253:d=9  hl=2 l=   3 prim: OBJECT            :X509v3 Basic Constraints
 1258:d=9  hl=2 l=   1 prim: BOOLEAN           :255
 1261:d=9  hl=2 l=   2 prim: OCTET STRING      [HEX DUMP]:3000
 1265:d=5  hl=2 l=  10 cons: SEQUENCE
 1267:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1277:d=5  hl=2 l= 103 prim: BIT STRING
 1382:d=3  hl=4 l= 332 cons: SET
 1386:d=4  hl=4 l= 328 cons: SEQUENCE
 1390:d=5  hl=2 l=   1 prim: INTEGER           :01
 1393:d=5  hl=2 l=  85 cons: SEQUENCE
 1395:d=6  hl=2 l=  77 cons: SEQUENCE
 1397:d=7  hl=2 l=  18 cons: SET
 1399:d=8  hl=2 l=  16 cons: SEQUENCE
 1401:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1413:d=9  hl=2 l=   2 prim: IA5STRING         :ca
 1417:d=7  hl=2 l=  25 cons: SET
 1419:d=8  hl=2 l=  23 cons: SEQUENCE
 1421:d=9  hl=2 l=  10 prim: OBJECT            :domainComponent
 1433:d=9  hl=2 l=   9 prim: IA5STRING         :sandelman
 1444:d=7  hl=2 l=  28 cons: SET
 1446:d=8  hl=2 l=  26 cons: SEQUENCE
 1448:d=9  hl=2 l=   3 prim: OBJECT            :commonName
 1453:d=9  hl=2 l=  19 prim: UTF8STRING        :Unstrung Highway CA
 1474:d=6  hl=2 l=   4 prim: INTEGER           :23CC8913
 1480:d=5  hl=2 l=  11 cons: SEQUENCE
 1482:d=6  hl=2 l=   9 prim: OBJECT            :sha256
 1493:d=5  hl=2 l= 105 cons: cont [ 0 ]
 1495:d=6  hl=2 l=  24 cons: SEQUENCE
 1497:d=7  hl=2 l=   9 prim: OBJECT            :contentType
 1508:d=7  hl=2 l=  11 cons: SET
 1510:d=8  hl=2 l=   9 prim: OBJECT            :pkcs7-data
 1521:d=6  hl=2 l=  28 cons: SEQUENCE
 1523:d=7  hl=2 l=   9 prim: OBJECT            :signingTime
 1534:d=7  hl=2 l=  15 cons: SET
 1536:d=8  hl=2 l=  13 prim: UTCTIME           :190516025142Z
 1551:d=6  hl=2 l=  47 cons: SEQUENCE
 1553:d=7  hl=2 l=   9 prim: OBJECT            :messageDigest
 1564:d=7  hl=2 l=  34 cons: SET
 1566:d=8  hl=2 l=  32 prim: OCTET STRING      [HEX DUMP]:98461E22DB5423
 1600:d=5  hl=2 l=  10 cons: SEQUENCE
```

```
 1602:d=6  hl=2 l=   8 prim: OBJECT            :ecdsa-with-SHA256
 1612:d=5  hl=2 l= 104 prim: OCTET STRING      [HEX DUMP]:30660231009860
```

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael C. Richardson
Sandelman Software Works

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

Toerless Eckert
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara 95050
USA

Email: tte+ietf@cs.fau.de

Michael H. Behringer

Email: Michael.H.Behringer@gmail.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Network Working Group C. Bormann
Internet-Draft Universitaet Bremen TZI
Intended status: Standards Track B. Carpenter, Ed.
Expires: January 8, 2018 Univ. of Auckland
B. Liu, Ed.
Huawei Technologies Co., Ltd
July 7, 2017

A Generic Autonomic Signaling Protocol (GRASP)
draft-ietf-anima-grasp-15

Abstract

This document specifies the GeneRic Autonomic Signaling Protocol (GRASP), which enables autonomic nodes and autonomic service agents to dynamically discover peers, to synchronize state with each other, and to negotiate parameter settings with each other. GRASP depends on an external security environment that is described elsewhere. The technical objectives and parameters for specific application scenarios are to be described in separate documents. Appendices briefly discuss requirements for the protocol and existing protocols with comparable features.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The success of the Internet has made IP-based networks bigger and more complicated. Large-scale ISP and enterprise networks have become more and more problematic for human based management. Also, operational costs are growing quickly. Consequently, there are increased requirements for autonomic behavior in the networks. General aspects of autonomic networks are discussed in [RFC7575] and [RFC7576].

One approach is to largely decentralize the logic of network management by migrating it into network elements. A reference model for autonomic networking on this basis is given in [I-D.ietf-anima-reference-model]. The reader should consult this document to understand how various autonomic components fit together. In order to fulfill autonomy, devices that embody Autonomic Service

Agents (ASAs, [RFC7575]) have specific signaling requirements. In particular they need to discover each other, to synchronize state with each other, and to negotiate parameters and resources directly with each other. There is no limitation on the types of parameters and resources concerned, which can include very basic information needed for addressing and routing, as well as anything else that might be configured in a conventional non-autonomic network. The atomic unit of discovery, synchronization or negotiation is referred to as a technical objective, i.e, a configurable parameter or set of parameters (defined more precisely in Section 2.1).

Negotiation is an iterative process, requiring multiple message exchanges forming a closed loop between the negotiating entities. In fact, these entities are ASAs, normally but not necessarily in different network devices. State synchronization, when needed, can be regarded as a special case of negotiation, without iteration. Both negotiation and synchronization must logically follow discovery. More details of the requirements are found in Appendix E. Section 2.3 describes a behavior model for a protocol intended to support discovery, synchronization and negotiation. The design of GeneRic Autonomic Signaling Protocol (GRASP) in Section 2 of this document is based on this behavior model. The relevant capabilities of various existing protocols are reviewed in Appendix F.

The proposed discovery mechanism is oriented towards synchronization and negotiation objectives. It is based on a neighbor discovery process on the local link, but also supports diversion to peers on other links. There is no assumption of any particular form of network topology. When a device starts up with no pre-configuration, it has no knowledge of the topology. The protocol itself is capable of being used in a small and/or flat network structure such as a small office or home network as well as in a large professionally managed network. Therefore, the discovery mechanism needs to be able to allow a device to bootstrap itself without making any prior assumptions about network structure.

Because GRASP can be used as part of a decision process among distributed devices or between networks, it must run in a secure and strongly authenticated environment.

In realistic deployments, not all devices will support GRASP. Therefore, some autonomic service agents will directly manage a group of non-autonomic nodes, and other non-autonomic nodes will be managed traditionally. Such mixed scenarios are not discussed in this specification.

2. GRASP Protocol Overview

2.1. 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 [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

This document uses terminology defined in [RFC7575].

The following additional terms are used throughout this document:

- Discovery: a process by which an ASA discovers peers according to a specific discovery objective. The discovery results may be different according to the different discovery objectives. The discovered peers may later be used as negotiation counterparts or as sources of synchronization data.

- Negotiation: a process by which two ASAs interact iteratively to agree on parameter settings that best satisfy the objectives of both ASAs.

- State Synchronization: a process by which ASAs interact to receive the current state of parameter values stored in other ASAs. This is a special case of negotiation in which information is sent but the ASAs do not request their peers to change parameter settings. All other definitions apply to both negotiation and synchronization.

- Technical Objective (usually abbreviated as Objective): A technical objective is a data structure, whose main contents are a name and a value. The value consists of a single configurable parameter or a set of parameters of some kind. The exact format of an objective is defined in Section 2.10.1. An objective occurs in three contexts: Discovery, Negotiation and Synchronization. Normally, a given objective will not occur in negotiation and synchronization contexts simultaneously.

  * One ASA may support multiple independent objectives.

  * The parameter(s) in the value of a given objective apply to a specific service or function or action. They may in principle be anything that can be set to a specific logical, numerical or string value, or a more complex data structure, by a network

node. Each node is expected to contain one or more ASAs which may each manage subsidiary non-autonomic nodes.

* Discovery Objective: an objective in the process of discovery. Its value may be undefined.

* Synchronization Objective: an objective whose specific technical content needs to be synchronized among two or more ASAs. Thus, each ASA will maintain its own copy of the objective.

* Negotiation Objective: an objective whose specific technical content needs to be decided in coordination with another ASA. Again, each ASA will maintain its own copy of the objective.

A detailed discussion of objectives, including their format, is found in Section 2.10.

- Discovery Initiator: an ASA that starts discovery by sending a discovery message referring to a specific discovery objective.

- Discovery Responder: a peer that either contains an ASA supporting the discovery objective indicated by the discovery initiator, or caches the locator(s) of the ASA(s) supporting the objective. It sends a Discovery Response, as described later.

- Synchronization Initiator: an ASA that starts synchronization by sending a request message referring to a specific synchronization objective.

- Synchronization Responder: a peer ASA which responds with the value of a synchronization objective.

- Negotiation Initiator: an ASA that starts negotiation by sending a request message referring to a specific negotiation objective.

- Negotiation Counterpart: a peer with which the Negotiation Initiator negotiates a specific negotiation objective.

- GRASP Instance: This refers to an instantiation of a GRASP protocol engine, likely including multiple threads or processes as well as dynamic data structures such as a discovery cache, running in a given security environment on a single device.

- GRASP Core: This refers to the code and shared data structures of a GRASP instance, which will communicate with individual ASAs via a suitable Application Programming Interface (API).

- o Interface or GRASP Interface: Unless otherwise stated, these refer to a network interface - which might be physical or virtual - that a specific instance of GRASP is currently using. A device might have other interfaces that are not used by GRASP and which are outside the scope of the autonomic network.

2.2. High Level Deployment Model

A GRASP implementation will be part of the Autonomic Networking Infrastructure (ANI) in an autonomic node, which must also provide an appropriate security environment. In accordance with [I-D.ietf-anima-reference-model], this SHOULD be the Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. As a result, all autonomic nodes in the ACP are able to trust each other. It is expected that GRASP will access the ACP by using a typical socket programming interface and the ACP will make available only network interfaces within the autonomic network. If there is no ACP, the considerations described in Section 2.5.1 apply.

There will also be one or more Autonomic Service Agents (ASAs). In the minimal case of a single-purpose device, these components might be fully integrated with GRASP and the ACP. A more common model is expected to be a multi-purpose device capable of containing several ASAs, such as a router or large switch. In this case it is expected that the ACP, GRASP and the ASAs will be implemented as separate processes, which are able to support asynchronous and simultaneous operations, for example by multi-threading.

In some scenarios, a limited negotiation model might be deployed based on a limited trust relationship such as that between two administrative domains. ASAs might then exchange limited information and negotiate some particular configurations.

GRASP is explicitly designed to operate within a single addressing realm. Its discovery and flooding mechanisms do not support autonomic operations that cross any form of address translator or upper layer proxy.

A suitable Application Programming Interface (API) will be needed between GRASP and the ASAs. In some implementations, ASAs would run in user space with a GRASP library providing the API, and this library would in turn communicate via system calls with core GRASP functions. Details of the API are out of scope for the present document. For further details of possible deployment models, see [I-D.ietf-anima-reference-model].

An instance of GRASP must be aware of the network interfaces it will use, and of the appropriate global-scope and link-local addresses.

In the presence of the ACP, such information will be available from the adjacency table discussed in [I-D.ietf-anima-reference-model]. In other cases, GRASP must determine such information for itself. Details depend on the device and operating system. In the rest of this document, the terms 'interfaces' or 'GRASP interfaces' refers only to the set of network interfaces that a specific instance of GRASP is currently using.

Because GRASP needs to work with very high reliability, especially during bootstrapping and during fault conditions, it is essential that every implementation continues to operate in adverse conditions. For example, discovery failures, or any kind of socket exception at any time, must not cause irrecoverable failures in GRASP itself, and must return suitable error codes through the API so that ASAs can also recover.

GRASP must not depend upon non-volatile data storage. All run time error conditions, and events such as address renumbering, network interface failures, and CPU sleep/wake cycles, must be handled in such a way that GRASP will still operate correctly and securely (Section 2.5.1) afterwards.

An autonomic node will normally run a single instance of GRASP, used by multiple ASAs. Possible exceptions are mentioned below.

2.3. High Level Design

This section describes the behavior model and general design of GRASP, supporting discovery, synchronization and negotiation, to act as a platform for different technical objectives.

- A generic platform:

  The protocol design is generic and independent of the synchronization or negotiation contents. The technical contents will vary according to the various technical objectives and the different pairs of counterparts.

- Normally, a single main instance of the GRASP protocol engine will exist in an autonomic node, and each ASA will run as an independent asynchronous process. However, scenarios where multiple instances of GRASP run in a single node, perhaps with different security properties, are possible (Section 2.5.2). In this case, each instance MUST listen independently for GRASP link-local multicasts, and all instances MUST be woken by each such multicast, in order for discovery and flooding to work correctly.

o Security infrastructure:

As noted above, the protocol itself has no built-in security functionality, and relies on a separate secure infrastructure.

o Discovery, synchronization and negotiation are designed together:

The discovery method and the synchronization and negotiation methods are designed in the same way and can be combined when this is useful, allowing a rapid mode of operation described in Section 2.5.4. These processes can also be performed independently when appropriate.

* Thus, for some objectives, especially those concerned with application layer services, another discovery mechanism such as the future DNS Service Discovery [RFC7558] MAY be used. The choice is left to the designers of individual ASAs.

o A uniform pattern for technical objectives:

The synchronization and negotiation objectives are defined according to a uniform pattern. The values that they contain could be carried either in a simple binary format or in a complex object format. The basic protocol design uses the Concise Binary Object Representation (CBOR) [RFC7049], which is readily extensible for unknown future requirements.

o A flexible model for synchronization:

GRASP supports synchronization between two nodes, which could be used repeatedly to perform synchronization among a small number of nodes. It also supports an unsolicited flooding mode when large groups of nodes, possibly including all autonomic nodes, need data for the same technical objective.

* There may be some network parameters for which a more traditional flooding mechanism such as DNCP [RFC7787] is considered more appropriate. GRASP can coexist with DNCP.

o A simple initiator/responder model for negotiation:

Multi-party negotiations are very complicated to model and cannot readily be guaranteed to converge. GRASP uses a simple bilateral model and can support multi-party negotiations by indirect steps.

o Organizing of synchronization or negotiation content:

The technical content transmitted by GRASP will be organized according to the relevant function or service. The objectives for different functions or services are kept separate, because they may be negotiated or synchronized with different counterparts or have different response times. Thus a normal arrangement would be a single ASA managing a small set of closely related objectives, with a version of that ASA in each relevant autonomic node. Further discussion of this aspect is out of scope for the current document.

o Requests and responses in negotiation procedures:

The initiator can negotiate a specific negotiation objective with relevant counterpart ASAs. It can request relevant information from a counterpart so that it can coordinate its local configuration. It can request the counterpart to make a matching configuration. It can request simulation or forecast results by sending some dry run conditions.

Beyond the traditional yes/no answer, the responder can reply with a suggested alternative value for the objective concerned. This would start a bi-directional negotiation ending in a compromise between the two ASAs.

o Convergence of negotiation procedures:

To enable convergence, when a responder suggests a new value or condition in a negotiation step reply, it should be as close as possible to the original request or previous suggestion. The suggested value of later negotiation steps should be chosen between the suggested values from the previous two steps. GRASP provides mechanisms to guarantee convergence (or failure) in a small number of steps, namely a timeout and a maximum number of iterations.

o Extensibility:

GRASP intentionally does not have a version number, and can be extended by adding new message types and options. The Invalid Message (M_INVALID) will be used to signal that an implementation does not recognize a message or option sent by another implementation. In normal use, new semantics will be added by defining new synchronization or negotiation objectives.

2.4. Quick Operating Overview

An instance of GRASP is expected to run as a separate core module, providing an API (such as [I-D.liu-anima-grasp-api]) to interface to various ASAs. These ASAs may operate without special privilege, unless they need it for other reasons (such as configuring IP addresses or manipulating routing tables).

The GRASP mechanisms used by the ASA are built around GRASP objectives 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 serialize it for transmission in CBOR, which is no restriction at all in practice.

GRASP provides the following mechanisms:

- A discovery mechanism (M_DISCOVERY, M_RESPONSE), by which an ASA can discover other ASAs supporting a given objective.
- A negotiation request mechanism (M_REQ_NEG), by which an ASA can start negotiation of an objective with a counterpart ASA. Once a negotiation has started, the process is symmetrical, and there is a negotiation step message (M_NEGOTIATE) for each ASA to use in turn. Two other functions support negotiating steps (M_WAIT, M_END).
- A synchronization mechanism (M_REQ_SYN), by which an ASA can request the current value of an objective from a counterpart ASA. With this, there is a corresponding response function (M_SYNCH) for an ASA that wishes to respond to synchronization requests.
- A flood mechanism (M_FLOOD), by which an ASA can cause the current value of an objective to be flooded throughout the autonomic network so that any ASA can receive it. One application of this is to act as an announcement, avoiding the need for discovery of a widely applicable objective.

Some example messages and simple message flows are provided in Appendix D.

2.5. GRASP Protocol Basic Properties and Mechanisms

2.5.1. Required External Security Mechanism

GRASP does not specify transport security because it is meant to be adapted to different environments. Every solution adopting GRASP MUST specify a security and transport substrate used by GRASP in that solution.

The substrate MUST enforce sending and receiving GRASP messages only between members of a mutually trusted group running GRASP. Each group member is an instance of GRASP. The group members are nodes of a connected graph. The group and graph is created by the security and transport substrate and called the GRASP domain. The substrate must support unicast messages between any group members and (link-local) multicast messages between adjacent group members. It must deny messages between group members and non group members. With this model, security is provided by enforcing group membership, but any member of the trusted group can attack the entire network until revoked.

Substrates MUST use cryptographic member authentication and message integrity for GRASP messages. This can be end-to-end or hop-by-hop across the domain. The security and transport substrate MUST provide mechanisms to remove untrusted members from the group.

If the substrate does not mandate and enforce GRASP message encryption then any service using GRASP in such a solution MUST provide protection and encryption for message elements whose exposure could constitute an attack vector.

The security and transport substrate for GRASP in the ANI is the ACP. Unless otherwise noted, we assume this security and transport substrate in the remainder of this document. The ACP does mandate the use of encryption; therefore GRASP in the ANI can rely on GRASP message being encrypted. The GRASP domain is the ACP: all nodes in an autonomic domain connected by encrypted virtual links formed by the ACP. The ACP uses hop-by-hop security (authentication/ encryption) of messages. Removal of nodes relies on standard PKI certificate revocation or expiry of sufficiently short lived certificates. Refer to [I-D.ietf-anima-autonomic-control-plane] for more details.

As mentioned in Section 2.3, some GRASP operations might be performed across an administrative domain boundary by mutual agreement, without the benefit of an ACP. Such operations MUST be confined to a separate instance of GRASP with its own copy of all GRASP data structures running across a separate GRASP domain with a security and transport substrate. In the most simple case, each point-to-point interdomain GRASP peering could be a separate domain and the security and transport substrate could be built using transport or network layer security protocols. This is subject to future specifications.

An exception to the requirements for the security and transport substrate exists for highly constrained subsets of GRASP meant to support the establishment of a security and transport substrate, described in the following section.

2.5.2. Discovery Unsolicited Link-Local (DULL) GRASP

Some services may need to use insecure GRASP discovery, response and flood messages without being able to use pre-existing security associations, for example as part of discovery for establishing security associations such as a security substrate for GRASP.

Such operations being intrinsically insecure, they need to be confined to link-local use to minimize the risk of malicious actions. Possible examples include discovery of candidate ACP neighbors [I-D.ietf-anima-autonomic-control-plane], discovery of bootstrap proxies [I-D.ietf-anima-bootstrapping-keyinfra] or perhaps initialization services in networks using GRASP without being fully autonomic (e.g., no ACP). Such usage MUST be limited to link-local operations on a single interface and MUST be confined to a separate insecure instance of GRASP with its own copy of all GRASP data structures. This instance is nicknamed DULL - Discovery Unsolicited Link-Local.

The detailed rules for the DULL instance of GRASP are as follows:

- o An initiator MAY send Discovery or Flood Synchronization link-local multicast messages which MUST have a loop count of 1, to prevent off-link operations. Other unsolicited GRASP message types MUST NOT be sent.

- o A responder MUST silently discard any message whose loop count is not 1.

- o A responder MUST silently discard any message referring to a GRASP Objective that is not directly part of a service that requires this insecure mode.

- o A responder MUST NOT relay any multicast messages.

- o A Discovery Response MUST indicate a link-local address.

- o A Discovery Response MUST NOT include a Divert option.

- o A node MUST silently discard any message whose source address is not link-local.

To minimize traffic possibly observed by third parties, GRASP traffic SHOULD be minimized by using only Flood Synchronization to announce objectives and their associated locators, rather than by using Discovery and Response. Further details are out of scope for this document

2.5.3. Transport Layer Usage

All GRASP messages, after they are serialized as a CBOR byte string, are transmitted as such directly over the transport protocol in use. The transport protocol(s) for a GRASP domain are specified by the security and transport substrate as introduced in Section 2.5.1.

GRASP discovery and flooding messages are designed for GRASP domain wide flooding through hop-by-hop link-local multicast forwarding between adjacent GRASP nodes. The GRASP security and transport substrate needs to specify how these link local multicasts are transported. This can be unreliable transport (UDP) but it SHOULD be reliable transport (e.g., TCP).

If the substrate specifies an unreliable transport such as UDP for discovery and flooding messages, then it MUST NOT use IP fragmentation because of its loss characteristic, especially in multi-hop flooding. GRASP MUST then enforce at the user API level a limit to the size of discovery and flooding messages, so that no fragmentation can occur. For IPv6 transport this means that those messages must be at most 1280 bytes sized IPv6 packets (unless there is a known larger minimum link MTU across the whole GRASP domain).

All other GRASP messages are unicast beteween group members of the GRASP domain. These MUST use a reliable transport protocol because GRASP itself does not provide for error detection, retransmission or flow control. Unless otherwise specified by the security and transport substrate, TCP MUST be used.

The security and transport substrate for GRASP in the ANI is the ACP. Unless otherwise noted, we assume this security and transport substrate in the remainder of this document when describing GRASPs message transport. In the ACP, TCP is used for GRASP unicast messages. GRASP discovery and flooding messages also use TCP: These link-local messages are forwarded by replicating them to all adjacent GRASP nodes on the link via TCP connections to those adjacent GRASP nodes. Because of this, GRASP in the ANI has no limitations on the size of discovery and flooding messages with respect to fragmentation issues. UDP is used in the ANI with GRASP only with DULL when the ACP is built to discover ACP/GRASP neighbors on links.

For link-local UDP multicast, the GRASP protocol listens to the well-known GRASP Listen Port (Section 2.6). Transport connections for Discovery and Flooding on relay nodes must terminate in GRASP instances (eg: GRASP ASAs) so that link-local multicast, hop-by-hop flooding of M_DISCOVERY and M_FLOOD and hop-by-hop forwarding of M_RESPONSE and caching of those responses along the path work correctly.

Unicast transport connections used for synchronization and negotiation can terminate directly in ASAs that implement objectives and therefore this traffic does not need to pass through GRASP instances. For this, the ASA listens on its own dynamically assigned ports, which are communicated to its peers during discovery. Alternatively, the GRASP instance can also terminate the unicast transport connections and pass the traffic from/to the ASA if that is preferrable in some implementation (eg: to better decouple ASAs from network connections).

2.5.4. Discovery Mechanism and Procedures

2.5.4.1. Separated discovery and negotiation mechanisms

Although discovery and negotiation or synchronization are defined together in GRASP, they are separate mechanisms. The discovery process could run independently from the negotiation or synchronization process. Upon receiving a Discovery (Section 2.8.4) message, the recipient node should return a response message in which it either indicates itself as a discovery responder or diverts the initiator towards another more suitable ASA. However, this response may be delayed if the recipient needs to relay the discovery onwards, as described below.

The discovery action (M_DISCOVERY) will normally be followed by a negotiation (M_REQ_NEG) or synchronization (M_REQ_SYN) action. The discovery results could be utilized by the negotiation protocol to decide which ASA the initiator will negotiate with.

The initiator of a discovery action for a given objective need not be capable of responding to that objective as a Negotiation Counterpart, as a Synchronization Responder or as source for flooding. For example, an ASA might perform discovery even if it only wishes to act a Synchronization Initiator or Negotiation Initiator. Such an ASA does not itself need to respond to discovery messages.

It is also entirely possible to use GRASP discovery without any subsequent negotiation or synchronization action. In this case, the discovered objective is simply used as a name during the discovery process and any subsequent operations between the peers are outside the scope of GRASP.

2.5.4.2. Discovery Overview

A complete discovery process will start with a multicast (of M_DISCOVERY) on the local link. On-link neighbors supporting the discovery objective will respond directly (with M_RESPONSE). A neighbor with multiple interfaces may respond with a cached discovery response. If it has no cached response, it will relay the discovery on its other GRASP interfaces. If a node receiving the relayed discovery supports the discovery objective, it will respond to the relayed discovery. If it has a cached response, it will respond with that. If not, it will repeat the discovery process, which thereby becomes iterative. The loop count and timeout will ensure that the process ends. Further details are given below.

A Discovery message MAY be sent unicast to a peer node, which SHOULD then proceed exactly as if the message had been multicast, except that when TCP is used, the response will be on the same socket as the query. However, this mode does not guarantee successful discovery in the general case.

2.5.4.3. Discovery Procedures

Discovery starts as an on-link operation. The Divert option can tell the discovery initiator to contact an off-link ASA for that discovery objective. If the security and transport substrate of the GRASP domain (see Section 2.5.3) uses UDP link-local multicast then the discovery initiator sends these to the ALL_GRASP_NEIGHBORS link-local multicast address (Section 2.6) and and all GRASP nodes need to listen to this address to act as discovery responder. Because this port is unique in a device, this is a function of the GRASP instance and not of an individual ASA. As a result, each ASA will need to register the objectives that it supports with the local GRASP instance.

If an ASA in a neighbor device supports the requested discovery objective, the device SHOULD respond to the link-local multicast with a unicast Discovery Response message (Section 2.8.5) with locator option(s), unless it is temporarily unavailable. Otherwise, if the neighbor has cached information about an ASA that supports the requested discovery objective (usually because it discovered the same objective before), it SHOULD respond with a Discovery Response message with a Divert option pointing to the appropriate Discovery

Responder. However, it SHOULD NOT respond with a cached response on an interface if it learnt that information from the same interface, because the peer in question will answer directly if still operational.

If a device has no information about the requested discovery objective, and is not acting as a discovery relay (see below) it MUST silently discard the Discovery message.

The discovery initiator MUST set a reasonable timeout on the discovery process. A suggested value is 100 milliseconds multiplied by the loop count embedded in the objective.

If no discovery response is received within the timeout, the Discovery message MAY be repeated, with a newly generated Session ID (Section 2.7). An exponential backoff SHOULD be used for subsequent repetitions, to limit the load during busy periods. The details of the backoff algorithm will depend on the use case for the objective concerned but MUST be consistent with the recommendations in [RFC8085] for low data-volume multicast. Frequent repetition might be symptomatic of a denial of service attack.

After a GRASP device successfully discovers a locator for a Discovery Responder supporting a specific objective, it SHOULD cache this information, including the interface index [RFC3493] via which it was discovered. This cache record MAY be used for future negotiation or synchronization, and the locator SHOULD be passed on when appropriate as a Divert option to another Discovery Initiator.

The cache mechanism MUST include a lifetime for each entry. The lifetime is derived from a time-to-live (ttl) parameter in each Discovery Response message. Cached entries MUST be ignored or deleted after their lifetime expires. In some environments, unplanned address renumbering might occur. In such cases, the lifetime SHOULD be short compared to the typical address lifetime. The discovery mechanism needs to track the node’s current address to ensure that Discovery Responses always indicate the correct address.

If multiple Discovery Responders are found for the same objective, they SHOULD all be cached, unless this creates a resource shortage. The method of choosing between multiple responders is an implementation choice. This choice MUST be available to each ASA but the GRASP implementation SHOULD provide a default choice.

Because Discovery Responders will be cached in a finite cache, they might be deleted at any time. In this case, discovery will need to be repeated. If an ASA exits for any reason, its locator might still be cached for some time, and attempts to connect to it will fail. ASAs need to be robust in these circumstances.

2.5.4.4. Discovery Relaying

A GRASP instance with multiple link-layer interfaces (typically running in a router) MUST support discovery on all GRASP interfaces. We refer to this as a 'relaying instance'.

DULL Instances (Section 2.5.2) are always single-interface instances and therefore MUST NOT perform discovery relaying.

If a relaying instance receives a Discovery message on a given interface for a specific objective that it does not support and for which it has not previously cached a Discovery Responder, it MUST relay the query by re-issuing a new Discovery message as a link-local multicast on its other GRASP interfaces.

The relayed discovery message MUST have the same Session ID and Initiator field as the incoming (see Section 2.8.4). The Initiator IP address field is only used to allow for disambiguation of the Session ID and is never used to address Response packets. Response packets are sent back to the relaying instance, not the original initiator.

The M_DISCOVERY message does not encode the transport address of the originator or relay. Response packets must therefore be sent to the transport layer address of the connection on which the M_DISCOVERY message was received. If the M_DISCOVERY was relayed via a reliable hop-by-hop transport connection, the response is simply sent back via the same connection.

If the M_DISCOVERY was relayed via link-local (eg: UDP) multicast, the response is sent back via a reliable hop-by-hop transport connection with the same port number as the source port of the link-local multicast. Therefore, if link-local multicast is used and M_RESPONSE messages are required (which is the case in almost all GRASP instances except for the limited use of DULL instances in the ANI), GRASP needs to be able to bind to one port number on UDP from which to originate the link-local multicast M_DISCOVERY messages and the same port number on the reliable hop-by-hop transport (eg: TCP by default) to be able to respond to transport connections from responders that want to send M_RESPONSE messages back. Note that this port does not need to be the GRASP_LISTEN_PORT.

The relaying instance MUST decrement the loop count within the objective, and MUST NOT relay the Discovery message if the result is zero. Also, it MUST limit the total rate at which it relays discovery messages to a reasonable value, in order to mitigate possible denial of service attacks. For example, the rate limit could be set to a small multiple of the observed rate of discovery messages during normal operation. The relaying instance MUST cache the Session ID value and initiator address of each relayed Discovery message until any Discovery Responses have arrived or the discovery process has timed out. To prevent loops, it MUST NOT relay a Discovery message which carries a given cached Session ID and initiator address more than once. These precautions avoid discovery loops and mitigate potential overload.

Since the relay device is unaware of the timeout set by the original initiator it SHOULD set a suitable timeout for the relayed discovery. A suggested value is 100 milliseconds multiplied by the remaining loop count.

The discovery results received by the relaying instance MUST in turn be sent as a Discovery Response message to the Discovery message that caused the relay action.

2.5.4.5. Rapid Mode (Discovery with Negotiation or Synchronization)

A Discovery message MAY include an Objective option. This allows a rapid mode of negotiation (Section 2.5.5.1) or synchronization (Section 2.5.6.3). Rapid mode is currently limited to a single objective for simplicity of design and implementation. A possible future extension is to allow multiple objectives in rapid mode for greater efficiency.

2.5.5. Negotiation Procedures

A negotiation initiator opens a transport connection to a counterpart ASA using the address, protocol and port obtained during discovery. It then sends a negotiation request (using M_REQ_NEG) to the counterpart, including a specific negotiation objective. It may request the negotiation counterpart to make a specific configuration. Alternatively, it may request a certain simulation or forecast result by sending a dry run configuration. The details, including the distinction between a dry run and a live configuration change, will be defined separately for each type of negotiation objective. Any state associated with a dry run operation, such as temporarily reserving a resource for subsequent use in a live run, is entirely a matter for the designer of the ASA concerned.

Each negotiation session as a whole is subject to a timeout (default GRASP_DEF_TIMEOUT milliseconds, Section 2.6), initialised when the request is sent (see Section 2.8.6). If no reply message of any kind is received within the timeout, the negotiation request MAY be repeated, with a newly generated Session ID (Section 2.7). An exponential backoff SHOULD be used for subsequent repetitions. The details of the backoff algorithm will depend on the use case for the objective concerned.

If the counterpart can immediately apply the requested configuration, it will give an immediate positive (O_ACCEPT) answer (using M_END). This will end the negotiation phase immediately. Otherwise, it will negotiate (using M_NEGOTIATE). It will reply with a proposed alternative configuration that it can apply (typically, a configuration that uses fewer resources than requested by the negotiation initiator). This will start a bi-directional negotiation (using M_NEGOTIATE) to reach a compromise between the two ASAs.

The negotiation procedure is ended when one of the negotiation peers sends a Negotiation Ending (M_END) message, which contains an accept (O_ACCEPT) or decline (O_DECLINE) option and does not need a response from the negotiation peer. Negotiation may also end in failure (equivalent to a decline) if a timeout is exceeded or a loop count is exceeded. When the procedure ends for whatever reason, the transport connection SHOULD be closed. A transport session failure is treated as a negotiation failure.

A negotiation procedure concerns one objective and one counterpart. Both the initiator and the counterpart may take part in simultaneous negotiations with various other ASAs, or in simultaneous negotiations about different objectives. Thus, GRASP is expected to be used in a multi-threaded mode or its logical equivalent. Certain negotiation objectives may have restrictions on multi-threading, for example to avoid over-allocating resources.

Some configuration actions, for example wavelength switching in optical networks, might take considerable time to execute. The ASA concerned needs to allow for this by design, but GRASP does allow for a peer to insert latency in a negotiation process if necessary (Section 2.8.9, M_WAIT).

2.5.5.1. Rapid Mode (Discovery/Negotiation Linkage)

A Discovery message MAY include a Negotiation Objective option. In this case it is as if the initiator sent the sequence M_DISCOVERY, immediately followed by M_REQ_NEG. This has implications for the construction of the GRASP core, as it must carefully pass the contents of the Negotiation Objective option to the ASA so that it may evaluate the objective directly. When a Negotiation Objective option is present the ASA replies with an M_NEGOTIATE message (or M_END with O_ACCEPT if it is immediately satisfied with the proposal), rather than with an M_RESPONSE. However, if the recipient node does not support rapid mode, discovery will continue normally.

It is possible that a Discovery Response will arrive from a responder that does not support rapid mode, before such a Negotiation message arrives. In this case, rapid mode will not occur.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. However, a network in which some nodes support rapid mode and others do not will have complex timing-dependent behaviors. Therefore, the rapid negotiation function SHOULD be disabled by default.

2.5.6. Synchronization and Flooding Procedures

2.5.6.1. Unicast Synchronization

A synchronization initiator opens a transport connection to a counterpart ASA using the address, protocol and port obtained during discovery. It then sends a synchronization request (using M_REQ_SYN) to the counterpart, including a specific synchronization objective. The counterpart responds with a Synchronization message (M_SYNCH, Section 2.8.10) containing the current value of the requested synchronization objective. No further messages are needed and the transport connection SHOULD be closed. A transport session failure is treated as a synchronization failure.

If no reply message of any kind is received within a given timeout (default GRASP_DEF_TIMEOUT milliseconds, Section 2.6), the synchronization request MAY be repeated, with a newly generated Session ID (Section 2.7). An exponential backoff SHOULD be used for subsequent repetitions. The details of the backoff algorithm will depend on the use case for the objective concerned.

2.5.6.2. Flooding

In the case just described, the message exchange is unicast and concerns only one synchronization objective. For large groups of nodes requiring the same data, synchronization flooding is available. For this, a flooding initiator MAY send an unsolicited Flood Synchronization message containing one or more Synchronization Objective option(s), if and only if the specification of those objectives permits it. This is sent as a multicast message to the ALL_GRASP_NEIGHBORS multicast address (Section 2.6).

Receiving flood multicasts is a function of the GRASP core, as in the case of discovery multicasts (Section 2.5.4.3).

To ensure that flooding does not result in a loop, the originator of the Flood Synchronization message MUST set the loop count in the objectives to a suitable value (the default is GRASP_DEF_LOOPCT). Also, a suitable mechanism is needed to avoid excessive multicast traffic. This mechanism MUST be defined as part of the specification of the synchronization objective(s) concerned. It might be a simple rate limit or a more complex mechanism such as the Trickle algorithm [RFC6206].

A GRASP device with multiple link-layer interfaces (typically a router) MUST support synchronization flooding on all GRASP interfaces. If it receives a multicast Flood Synchronization message on a given interface, it MUST relay it by re-issuing a Flood Synchronization message as a link-local multicast on its other GRASP interfaces. The relayed message MUST have the same Session ID as the incoming message and MUST be tagged with the IP address of its original initiator.

Link-layer Flooding is supported by GRASP by setting the loop count to 1, and sending with a link-local source address. Floods with link-local source addresses and a loop count other than 1 are invalid, and such messages MUST be discarded.

The relaying device MUST decrement the loop count within the first objective, and MUST NOT relay the Flood Synchronization message if the result is zero. Also, it MUST limit the total rate at which it relays Flood Synchronization messages to a reasonable value, in order to mitigate possible denial of service attacks. For example, the rate limit could be set to a small multiple of the observed rate of flood messages during normal operation. The relaying device MUST cache the Session ID value and initiator address of each relayed Flood Synchronization message for a time not less than twice GRASP_DEF_TIMEOUT milliseconds. To prevent loops, it MUST NOT relay a Flood Synchronization message which carries a given cached Session ID and initiator address more than once. These precautions avoid synchronization loops and mitigate potential overload.

Note that this mechanism is unreliable in the case of sleeping nodes, or new nodes that join the network, or nodes that rejoin the network after a fault. An ASA that initiates a flood SHOULD repeat the flood at a suitable frequency, which MUST be consistent with the recommendations in [RFC8085] for low data-volume multicast. The ASA SHOULD also act as a synchronization responder for the objective(s) concerned. Thus nodes that require an objective subject to flooding can either wait for the next flood or request unicast synchronization for that objective.

The multicast messages for synchronization flooding are subject to the security rules in Section 2.5.1. In practice this means that they MUST NOT be transmitted and MUST be ignored on receipt unless there is an operational ACP or equivalent strong security in place. However, because of the security weakness of link-local multicast (Section 4), synchronization objectives that are flooded SHOULD NOT contain unencrypted private information and SHOULD be validated by the recipient ASA.

2.5.6.3. Rapid Mode (Discovery/Synchronization Linkage)

A Discovery message MAY include a Synchronization Objective option. In this case the Discovery message also acts as a Request Synchronization message to indicate to the Discovery Responder that it could directly reply to the Discovery Initiator with a Synchronization message Section 2.8.10 with synchronization data for rapid processing, if the discovery target supports the corresponding synchronization objective. The design implications are similar to those discussed in Section 2.5.5.1.

It is possible that a Discovery Response will arrive from a responder that does not support rapid mode, before such a Synchronization message arrives. In this case, rapid mode will not occur.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. However, a network in which some nodes support rapid mode and others do not will have complex timing-dependent behaviors. Therefore, the rapid synchronization function SHOULD be configured off by default and MAY be configured on or off by Intent.

2.6. GRASP Constants

o ALL_GRASP_NEIGHBORS

  A link-local scope multicast address used by a GRASP-enabled device to discover GRASP-enabled neighbor (i.e., on-link) devices. All devices that support GRASP are members of this multicast group.

  * IPv6 multicast address: TBD1

  * IPv4 multicast address: TBD2

o GRASP_LISTEN_PORT (TBD3)

  A well-known UDP user port that every GRASP-enabled network device MUST listen to for link-local multicasts when UDP is used for

M_DISCOVERY or M_FLOOD messages in the GRASP instance This user port MAY also be used to listen for TCP or UDP unicast messages in a simple implementation of GRASP (Section 2.5.3).

o GRASP_DEF_TIMEOUT (60000 milliseconds)

  The default timeout used to determine that an operation has failed to complete.

o GRASP_DEF_LOOPCT (6)

  The default loop count used to determine that a negotiation has failed to complete, and to avoid looping messages.

o GRASP_DEF_MAX_SIZE (2048)

  The default maximum message size in bytes.

2.7. Session Identifier (Session ID)

This is an up to 32-bit opaque value used to distinguish multiple sessions between the same two devices. A new Session ID MUST be generated by the initiator for every new Discovery, Flood Synchronization or Request message. All responses and follow-up messages in the same discovery, synchronization or negotiation procedure MUST carry the same Session ID.

The Session ID SHOULD have a very low collision rate locally. It MUST be generated by a pseudo-random number generator (PRNG) using a locally generated seed which is unlikely to be used by any other device in the same network. The PRNG SHOULD be cryptographically strong [RFC4086]. When allocating a new Session ID, GRASP MUST check that the value is not already in use and SHOULD check that it has not been used recently, by consulting a cache of current and recent sessions. In the unlikely event of a clash, GRASP MUST generate a new value.

However, there is a finite probability that two nodes might generate the same Session ID value. For that reason, when a Session ID is communicated via GRASP, the receiving node MUST tag it with the initiator’s IP address to allow disambiguation. In the highly unlikely event of two peers opening sessions with the same Session ID value, this tag will allow the two sessions to be distinguished. Multicast GRASP messages and their responses, which may be relayed between links, therefore include a field that carries the initiator’s global IP address.

There is a highly unlikely race condition in which two peers start simultaneous negotiation sessions with each other using the same Session ID value. Depending on various implementation choices, this might lead to the two sessions being confused. See Section 2.8.6 for details of how to avoid this.

2.8. GRASP Messages

2.8.1. Message Overview

This section defines the GRASP message format and message types. Message types not listed here are reserved for future use.

The messages currently defined are:

Discovery and Discovery Response (M_DISCOVERY, M_RESPONSE).

Request Negotiation, Negotiation, Confirm Waiting and Negotiation End (M_REQ_NEG, M_NEGOTIATE, M_WAIT, M_END).

Request Synchronization, Synchronization, and Flood Synchronization (M_REQ_SYN, M_SYNCH, M_FLOOD.

No Operation and Invalid (M_NOOP, M_INVALID).

2.8.2. GRASP Message Format

GRASP messages share an identical header format and a variable format area for options. GRASP message headers and options are transmitted in Concise Binary Object Representation (CBOR) [RFC7049]. In this specification, they are described using CBOR data definition language (CDDL) [I-D.greevenbosch-appsawg-cbor-cddl]. Fragmentary CDDL is used to describe each item in this section. A complete and normative CDDL specification of GRASP is given in Section 5, including constants such as message types.

Every GRASP message, except the No Operation message, carries a Session ID (Section 2.7). Options are then presented serially in the options field.

In fragmentary CDDL, every GRASP message follows the pattern:

```
grasp-message = (message .within message-structure) / noop-message

message-structure = [MESSAGE_TYPE, session-id, ?initiator,
                     *grasp-option]

MESSAGE_TYPE = 1..255
session-id = 0..4294967295 ;up to 32 bits
grasp-option = any
```

The MESSAGE_TYPE indicates the type of the message and thus defines the expected options. Any options received that are not consistent with the MESSAGE_TYPE SHOULD be silently discarded.

The No Operation (noop) message is described in Section 2.8.13.

The various MESSAGE_TYPE values are defined in Section 5.

All other message elements are described below and formally defined in Section 5.

If an unrecognized MESSAGE_TYPE is received in a unicast message, an Invalid message (Section 2.8.12) MAY be returned. Otherwise the message MAY be logged and MUST be discarded. If an unrecognized MESSAGE_TYPE is received in a multicast message, it MAY be logged and MUST be silently discarded.

2.8.3. Message Size

GRASP nodes MUST be able to receive unicast messages of at least GRASP_DEF_MAX_SIZE bytes. GRASP nodes MUST NOT send unicast messages longer than GRASP_DEF_MAX_SIZE bytes unless a longer size is explicitly allowed for the objective concerned. For example, GRASP negotiation itself could be used to agree on a longer message size.

The message parser used by GRASP should be configured to know about the GRASP_DEF_MAX_SIZE, or any larger negotiated message size, so that it may defend against overly long messages.

The maximum size of multicast messages (M_DISCOVERY and M_FLOOD) depends on the link layer technology or link adaptation layer in use.

2.8.4. Discovery Message

In fragmentary CDDL, a Discovery message follows the pattern:

```
discovery-message = [M_DISCOVERY, session-id, initiator, objective]
```

A discovery initiator sends a Discovery message to initiate a discovery process for a particular objective option.

The discovery initiator sends all Discovery messages via UDP to port GRASP_LISTEN_PORT at the link-local ALL_GRASP_NEIGHBORS multicast address on each link-layer interface in use by GRASP. It then listens for unicast TCP responses on a given port, and stores the discovery results (including responding discovery objectives and corresponding unicast locators).

The listening port used for TCP MUST be the same port as used for sending the Discovery UDP multicast, on a given interface. In an implementation with a single GRASP instance in a node this MAY be GRASP_LISTEN_PORT. To support multiple instances in the same node, the GRASP discovery mechanism in each instance needs to find, for each interface, a dynamic port that it can bind to for both sending UDP link-local multicast and listening for TCP, before initiating any discovery.

The 'initiator' field in the message is a globally unique IP address of the initiator, for the sole purpose of disambiguating the Session ID in other nodes. If for some reason the initiator does not have a globally unique IP address, it MUST use a link-local address for this purpose that is highly likely to be unique, for example using [RFC7217]. Determination of a node's globally unique IP address is implementation-dependent.

A Discovery message MUST include exactly one of the following:

o a discovery objective option (Section 2.10.1). Its loop count MUST be set to a suitable value to prevent discovery loops (default value is GRASP_DEF_LOOPCT). If the discovery initiator requires only on-link responses, the loop count MUST be set to 1.

o a negotiation objective option (Section 2.10.1). This is used both for the purpose of discovery and to indicate to the discovery target that it MAY directly reply to the discovery initiatior with a Negotiation message for rapid processing, if it could act as the corresponding negotiation counterpart. The sender of such a Discovery message MUST initialize a negotiation timer and loop count in the same way as a Request Negotiation message (Section 2.8.6).

o a synchronization objective option (Section 2.10.1). This is used both for the purpose of discovery and to indicate to the discovery target that it MAY directly reply to the discovery initiator with a Synchronization message for rapid processing, if it could act as the corresponding synchronization counterpart. Its loop count

MUST be set to a suitable value to prevent discovery loops (default value is GRASP_DEF_LOOPCT).

As mentioned in Section 2.5.4.2, a Discovery message MAY be sent unicast to a peer node, which SHOULD then proceed exactly as if the message had been multicast.

2.8.5. Discovery Response Message

In fragmentary CDDL, a Discovery Response message follows the pattern:

```
response-message = [M_RESPONSE, session-id, initiator, ttl,
                   (+locator-option // divert-option), ?objective)]

ttl = 0..4294967295 ; in milliseconds
```

A node which receives a Discovery message SHOULD send a Discovery Response message if and only if it can respond to the discovery.

It MUST contain the same Session ID and initiator as the Discovery message.

It MUST contain a time-to-live (ttl) for the validity of the response, given as a positive integer value in milliseconds. Zero implies a value significantly greater than GRASP_DEF_TIMEOUT milliseconds (Section 2.6). A suggested value is ten times that amount.

It MAY include a copy of the discovery objective from the Discovery message.

It is sent to the sender of the Discovery message via TCP at the port used to send the Discovery message (as explained in Section 2.8.4). In the case of a relayed Discovery message, the Discovery Response is thus sent to the relay, not the original initiator.

In all cases, the transport session SHOULD be closed after sending the Discovery Response. A transport session failure is treated as no response.

If the responding node supports the discovery objective of the discovery, it MUST include at least one kind of locator option (Section 2.9.5) to indicate its own location. A sequence of multiple kinds of locator options (e.g. IP address option and FQDN option) is also valid.

If the responding node itself does not support the discovery objective, but it knows the locator of the discovery objective, then it SHOULD respond to the discovery message with a divert option (Section 2.9.2) embedding a locator option or a combination of multiple kinds of locator options which indicate the locator(s) of the discovery objective.

More details on the processing of Discovery Responses are given in Section 2.5.4.

2.8.6. Request Messages

In fragmentary CDDL, Request Negotiation and Request Synchronization messages follow the patterns:

```
request-negotiation-message = [M_REQ_NEG, session-id, objective]

request-synchronization-message = [M_REQ_SYN, session-id, objective]
```

A negotiation or synchronization requesting node sends the appropriate Request message to the unicast address of the negotiation or synchronization counterpart, using the appropriate protocol and port numbers (selected from the discovery result). If the discovery result is an FQDN, it will be resolved first.

A Request message MUST include the relevant objective option. In the case of Request Negotiation, the objective option MUST include the requested value.

When an initiator sends a Request Negotiation message, it MUST initialize a negotiation timer for the new negotiation thread. The default is GRASP_DEF_TIMEOUT milliseconds. Unless this timeout is modified by a Confirm Waiting message (Section 2.8.9), the initiator will consider that the negotiation has failed when the timer expires.

Similarly, when an initiator sends a Request Synchronization, it SHOULD initialize a synchronization timer. The default is GRASP_DEF_TIMEOUT milliseconds. The initiator will consider that synchronization has failed if there is no response before the timer expires.

When an initiator sends a Request message, it MUST initialize the loop count of the objective option with a value defined in the specification of the option or, if no such value is specified, with GRASP_DEF_LOOPCT.

If a node receives a Request message for an objective for which no ASA is currently listening, it MUST immediately close the relevant socket to indicate this to the initiator. This is to avoid unnecessary timeouts if, for example, an ASA exits prematurely but the GRASP core is listening on its behalf.

To avoid the highly unlikely race condition in which two nodes simultaneously request sessions with each other using the same Session ID (Section 2.7), when a node receives a Request message, it MUST verify that the received Session ID is not already locally active. In case of a clash, it MUST discard the Request message, in which case the initiator will detect a timeout.

2.8.7. Negotiation Message

In fragmentary CDDL, a Negotiation message follows the pattern:

```
negotiate-message = [M_NEGOTIATE, session-id, objective]
```

A negotiation counterpart sends a Negotiation message in response to a Request Negotiation message, a Negotiation message, or a Discovery message in Rapid Mode. A negotiation process MAY include multiple steps.

The Negotiation message MUST include the relevant Negotiation Objective option, with its value updated according to progress in the negotiation. The sender MUST decrement the loop count by 1. If the loop count becomes zero the message MUST NOT be sent. In this case the negotiation session has failed and will time out.

2.8.8. Negotiation End Message

In fragmentary CDDL, a Negotiation End message follows the pattern:

```
end-message = [M_END, session-id, accept-option / decline-option]
```

A negotiation counterpart sends an Negotiation End message to close the negotiation. It MUST contain either an accept or a decline option, defined in Section 2.9.3 and Section 2.9.4. It could be sent either by the requesting node or the responding node.

2.8.9. Confirm Waiting Message

In fragmentary CDDL, a Confirm Waiting message follows the pattern:

```
wait-message = [M_WAIT, session-id, waiting-time]
waiting-time = 0..4294967295 ; in milliseconds
```

A responding node sends a Confirm Waiting message to ask the requesting node to wait for a further negotiation response. It might be that the local process needs more time or that the negotiation depends on another triggered negotiation. This message MUST NOT include any other options. When received, the waiting time value overwrites and restarts the current negotiation timer (Section 2.8.6).

The responding node SHOULD send a Negotiation, Negotiation End or another Confirm Waiting message before the negotiation timer expires. If not, when the initiator's timer expires, the initiator MUST treat the negotiation procedure as failed.

2.8.10. Synchronization Message

In fragmentary CDDL, a Synchronization message follows the pattern:

```
synch-message = [M_SYNCH, session-id, objective]
```

A node which receives a Request Synchronization, or a Discovery message in Rapid Mode, sends back a unicast Synchronization message with the synchronization data, in the form of a GRASP Option for the specific synchronization objective present in the Request Synchronization.

2.8.11. Flood Synchronization Message

In fragmentary CDDL, a Flood Synchronization message follows the pattern:

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

ttl = 0..4294967295 ; in milliseconds
```

A node MAY initiate flooding by sending an unsolicited Flood Synchronization Message with synchronization data. This MAY be sent to port GRASP_LISTEN_PORT at the link-local ALL_GRASP_NEIGHBORS multicast address, in accordance with the rules in Section 2.5.6.

The initiator address is provided, as described for Discovery messages (Section 2.8.4), only to disambiguate the Session ID.

The message MUST contain a time-to-live (ttl) for the validity of the contents, given as a positive integer value in milliseconds. There is no default; zero indicates an indefinite lifetime.

The synchronization data are in the form of GRASP Option(s) for specific synchronization objective(s). The loop count(s) MUST be set to a suitable value to prevent flood loops (default value is GRASP_DEF_LOOPCT).

Each objective option MAY be followed by a locator option associated with the flooded objective. In its absence, an empty option MUST be included to indicate a null locator.

A node that receives a Flood Synchronization message MUST cache the received objectives for use by local ASAs. Each cached objective MUST be tagged with the locator option sent with it, or with a null tag if an empty locator option was sent. If a subsequent Flood Synchronization message carrying an objective with same name and the same tag, the corresponding cached copy of the objective MUST be overwritten. If a subsequent Flood Synchronization message carrying an objective with same name arrives with a different tag, a new cached entry MUST be created.

Note: the purpose of this mechanism is to allow the recipient of flooded values to distinguish between different senders of the same objective, and if necessary communicate with them using the locator, protocol and port included in the locator option. Many objectives will not need this mechanism, so they will be flooded with a null locator.

Cached entries MUST be ignored or deleted after their lifetime expires.

2.8.12. Invalid Message

In fragmentary CDDL, an Invalid message follows the pattern:

```
invalid-message = [M_INVALID, session-id, ?any]
```

This message MAY be sent by an implementation in response to an incoming unicast message that it considers invalid. The session-id MUST be copied from the incoming message. The content SHOULD be diagnostic information such as a partial copy of the invalid message up to the maximum message size. An M_INVALID message MAY be silently ignored by a recipient. However, it could be used in support of extensibility, since it indicates that the remote node does not support a new or obsolete message or option.

An M_INVALID message MUST NOT be sent in response to an M_INVALID message.

2.8.13. No Operation Message

In fragmentary CDDL, a No Operation message follows the pattern:

```
noop-message = [M_NOOP]
```

This message MAY be sent by an implementation that for practical reasons needs to initialize a socket. It MUST be silently ignored by a recipient.

2.9. GRASP Options

This section defines the GRASP options for the negotiation and synchronization protocol signaling. Additional options may be defined in the future.

2.9.1. Format of GRASP Options

GRASP options are CBOR objects that MUST start with an unsigned integer identifying the specific option type carried in this option. These option types are formally defined in Section 5. Apart from that the only format requirement is that each option MUST be a well-formed CBOR object. In general a CBOR array format is RECOMMENDED to limit overhead.

GRASP options may be defined to include encapsulated GRASP options.

2.9.2. Divert Option

The Divert option is used to redirect a GRASP request to another node, which may be more appropriate for the intended negotiation or synchronization. It may redirect to an entity that is known as a specific negotiation or synchronization counterpart (on-link or off-link) or a default gateway. The divert option MUST only be encapsulated in Discovery Response messages. If found elsewhere, it SHOULD be silently ignored.

A discovery initiator MAY ignore a Divert option if it only requires direct discovery responses.

In fragmentary CDDL, the Divert option follows the pattern:

```
divert-option = [O_DIVERT, +locator-option]
```

The embedded Locator Option(s) (Section 2.9.5) point to diverted destination target(s) in response to a Discovery message.

2.9.3. Accept Option

The accept option is used to indicate to the negotiation counterpart that the proposed negotiation content is accepted.

The accept option MUST only be encapsulated in Negotiation End messages. If found elsewhere, it SHOULD be silently ignored.

In fragmentary CDDL, the Accept option follows the pattern:

```
accept-option = [O_ACCEPT]
```

2.9.4. Decline Option

The decline option is used to indicate to the negotiation counterpart the proposed negotiation content is declined and end the negotiation process.

The decline option MUST only be encapsulated in Negotiation End messages. If found elsewhere, it SHOULD be silently ignored.

In fragmentary CDDL, the Decline option follows the pattern:

```
decline-option = [O_DECLINE, ?reason]
reason = text  ;optional UTF-8 error message
```

Note: there might be scenarios where an ASA wants to decline the proposed value and restart the negotiation process. In this case it is an implementation choice whether to send a Decline option or to continue with a Negotiate message, with an objective option that contains a null value, or one that contains a new value that might achieve convergence.

2.9.5. Locator Options

These locator options are used to present reachability information for an ASA, a device or an interface. They are Locator IPv6 Address Option, Locator IPv4 Address Option, Locator FQDN (Fully Qualified Domain Name) Option and URI (Uniform Resource Identifier) Option.

Since ASAs will normally run as independent user programs, locator options need to indicate the network layer locator plus the transport protocol and port number for reaching the target. For this reason, the Locator Options for IP addresses and FQDNs include this information explicitly. In the case of the URI Option, this information can be encoded in the URI itself.

Note: It is assumed that all locators used in locator options are in scope throughout the GRASP domain. As stated in Section 2.2, GRASP is not intended to work across disjoint addressing or naming realms.

2.9.5.1. Locator IPv6 address option

In fragmentary CDDL, the IPv6 address option follows the pattern:

```
ipv6-locator-option = [O_IPv6_LOCATOR, ipv6-address,
                       transport-proto, port-number]
ipv6-address = bytes .size 16

transport-proto = IPPROTO_TCP / IPPROTO_UDP
IPPROTO_TCP = 6
IPPROTO_UDP = 17
port-number = 0..65535
```

The content of this option is a binary IPv6 address followed by the protocol number and port number to be used.

Note 1: The IPv6 address MUST normally have global scope. However, during initialization, a link-local address MAY be used for specific objectives only (Section 2.5.2). In this case the corresponding Discovery Response message MUST be sent via the interface to which the link-local address applies.

Note 2: A link-local IPv6 address MUST NOT be used when this option is included in a Divert option.

Note 3: The IPPROTO values are taken from the existing IANA Protocol Numbers registry in order to specify TCP or UDP. If GRASP requires future values that are not in that registry, a new registry for values outside the range 0..255 will be needed.

2.9.5.2. Locator IPv4 address option

In fragmentary CDDL, the IPv4 address option follows the pattern:

```
ipv4-locator-option = [O_IPv4_LOCATOR, ipv4-address,
                       transport-proto, port-number]
ipv4-address = bytes .size 4
```

The content of this option is a binary IPv4 address followed by the protocol number and port number to be used.

Note: If an operator has internal network address translation for IPv4, this option MUST NOT be used within the Divert option.

2.9.5.3. Locator FQDN option

In fragmentary CDDL, the FQDN option follows the pattern:

```
fqdn-locator-option = [O_FQDN_LOCATOR, text,
                       transport-proto, port-number]
```

The content of this option is the Fully Qualified Domain Name of the target followed by the protocol number and port number to be used.

Note 1: Any FQDN which might not be valid throughout the network in question, such as a Multicast DNS name [RFC6762], MUST NOT be used when this option is used within the Divert option.

Note 2: Normal GRASP operations are not expected to use this option. It is intended for special purposes such as discovering external services.

2.9.5.4. Locator URI option

In fragmentary CDDL, the URI option follows the pattern:

```
uri-locator = [O_URI_LOCATOR, text,
               transport-proto / null, port-number / null]
```

The content of this option is the Uniform Resource Identifier of the target followed by the protocol number and port number to be used (or by null values if not required) [RFC3986].

Note 1: Any URI which might not be valid throughout the network in question, such as one based on a Multicast DNS name [RFC6762], MUST NOT be used when this option is used within the Divert option.

Note 2: Normal GRASP operations are not expected to use this option. It is intended for special purposes such as discovering external services. Therefore its use is not further described in this specification.

2.10. Objective Options

2.10.1. Format of Objective Options

An objective option is used to identify objectives for the purposes of discovery, negotiation or synchronization. All objectives MUST be in the following format, described in fragmentary CDDL:

```
objective = [objective-name, objective-flags, loop-count, ?objective-value]

objective-name = text
objective-value = any
loop-count = 0..255
```

All objectives are identified by a unique name which is a UTF-8 string [RFC3629], to be compared byte by byte.

The names of generic objectives MUST NOT include a colon (":") and MUST be registered with IANA (Section 6).

The names of privately defined objectives MUST include at least one colon (":"). The string preceding the last colon in the name MUST be globally unique and in some way identify the entity or person defining the objective. The following three methods MAY be used to create such a globally unique string:

1. The unique string is a decimal number representing a registered 32 bit Private Enterprise Number (PEN) [RFC5612] that uniquely identifies the enterprise defining the objective.

2. The unique string is a fully qualified domain name that uniquely identifies the entity or person defining the objective.

3. The unique string is an email address that uniquely identifies the entity or person defining the objective.

The GRASP protocol treats the objective name as an opaque string. For example, "EX1", "32473:EX1", "example.com:EX1", "example.org:EX1 and "user@example.org:EX1" would be five different objectives.

The 'objective-flags' field is described below.

The 'loop-count' field is used for terminating negotiation as described in Section 2.8.7. It is also used for terminating discovery as described in Section 2.5.4, and for terminating flooding as described in Section 2.5.6.2. It is placed in the objective rather than in the GRASP message format because, as far as the ASA is concerned, it is a property of the objective itself.

The 'objective-value' field is to express the actual value of a negotiation or synchronization objective. Its format is defined in the specification of the objective and may be a simple value or a data structure of any kind, as long as it can be represented in CBOR. It is optional because it is optional in a Discovery or Discovery Response message.

2.10.2. Objective flags

An objective may be relevant for discovery only, for discovery and negotiation, or for discovery and synchronization. This is expressed in the objective by logical flag bits:

```
objective-flags = uint .bits objective-flag
objective-flag = &(
F_DISC: 0    ; valid for discovery
F_NEG: 1     ; valid for negotiation
F_SYNCH: 2   ; valid for synchronization
F_NEG_DRY: 3 ; negotiation is dry-run
)
```

These bits are independent and may be combined appropriately, e.g. (F_DISC and F_SYNCH) or (F_DISC and F_NEG) or (F_DISC and F_NEG and F_NEG_DRY).

Note that for a given negotiation session, an objective must be either used for negotiation, or for dry-run negotiation. Mixing the two modes in a single negotiation is not possible.

2.10.3. General Considerations for Objective Options

As mentioned above, Objective Options MUST be assigned a unique name. As long as privately defined Objective Options obey the rules above, this document does not restrict their choice of name, but the entity or person concerned SHOULD publish the names in use.

Names are expressed as UTF-8 strings for convenience in designing Objective Options for localized use. For generic usage, names expressed in the ASCII subset of UTF-8 are RECOMMENDED. Designers planning to use non-ASCII names are strongly advised to consult [RFC7564] or its successor to understand the complexities involved. Since the GRASP protocol compares names byte by byte, all issues of Unicode profiling and canonicalization MUST be specified in the design of the Objective Option.

All Objective Options MUST respect the CBOR patterns defined above as "objective" and MUST replace the "any" field with a valid CBOR data definition for the relevant use case and application.

An Objective Option that contains no additional fields beyond its "loop-count" can only be a discovery objective and MUST only be used in Discovery and Discovery Response messages.

The Negotiation Objective Options contain negotiation objectives, which vary according to different functions/services. They MUST be carried by Discovery, Request Negotiation or Negotiation messages only. The negotiation initiator MUST set the initial "loop-count" to a value specified in the specification of the objective or, if no such value is specified, to GRASP_DEF_LOOPCT.

For most scenarios, there should be initial values in the negotiation requests. Consequently, the Negotiation Objective options MUST always be completely presented in a Request Negotiation message, or in a Discovery message in rapid mode. If there is no initial value, the value field SHOULD be set to the ’null’ value defined by CBOR.

Synchronization Objective Options are similar, but MUST be carried by Discovery, Discovery Response, Request Synchronization, or Flood Synchronization messages only. They include value fields only in Synchronization or Flood Synchronization messages.

The design of an objective interacts in various ways with the design of the ASAs that will use it. ASA design considerations are discussed in [I-D.carpenter-anima-asa-guidelines].

2.10.4. Organizing of Objective Options

Generic objective options MUST be specified in documents available to the public and SHOULD be designed to use either the negotiation or the synchronization mechanism described above.

As noted earlier, one negotiation objective is handled by each GRASP negotiation thread. Therefore, a negotiation objective, which is based on a specific function or action, SHOULD be organized as a single GRASP option. It is NOT RECOMMENDED to organize multiple negotiation objectives into a single option, nor to split a single function or action into multiple negotiation objectives.

It is important to understand that GRASP negotiation does not support transactional integrity. If transactional integrity is needed for a specific objective, this must be ensured by the ASA. For example, an ASA might need to ensure that it only participates in one negotiation thread at the same time. Such an ASA would need to stop listening for incoming negotiation requests before generating an outgoing negotiation request.

A synchronization objective SHOULD be organized as a single GRASP option.

Some objectives will support more than one operational mode. An example is a negotiation objective with both a "dry run" mode (where the negotiation is to find out whether the other end can in fact make the requested change without problems) and a "live" mode, as explained in Section 2.5.5. The semantics of such modes will be defined in the specification of the objectives. These objectives SHOULD include flags indicating the applicable mode(s).

An issue requiring particular attention is that GRASP itself is not a transactionally safe protocol. Any state associated with a dry run operation, such as temporarily reserving a resource for subsequent use in a live run, is entirely a matter for the designer of the ASA concerned.

As indicated in Section 2.1, an objective's value may include multiple parameters. Parameters might be categorized into two classes: the obligatory ones presented as fixed fields; and the optional ones presented in some other form of data structure embedded in CBOR. The format might be inherited from an existing management or configuration protocol, with the objective option acting as a carrier for that format. The data structure might be defined in a formal language, but that is a matter for the specifications of individual objectives. There are many candidates, according to the context, such as ABNF, RBNF, XML Schema, YANG, etc. The GRASP protocol itself is agnostic on these questions. The only restriction is that the format can be mapped into CBOR.

It is NOT RECOMMENDED to mix parameters that have significantly different response time characteristics in a single objective. Separate objectives are more suitable for such a scenario.

All objectives MUST support GRASP discovery. However, as mentioned in Section 2.3, it is acceptable for an ASA to use an alternative method of discovery.

Normally, a GRASP objective will refer to specific technical parameters as explained in Section 2.1. However, it is acceptable to define an abstract objective for the purpose of managing or coordinating ASAs. It is also acceptable to define a special-purpose objective for purposes such as trust bootstrapping or formation of the ACP.

To guarantee convergence, a limited number of rounds or a timeout is needed for each negotiation objective. Therefore, the definition of each negotiation objective SHOULD clearly specify this, for example a default loop count and timeout, so that the negotiation can always be terminated properly. If not, the GRASP defaults will apply.

There must be a well-defined procedure for concluding that a negotiation cannot succeed, and if so deciding what happens next (e.g., deadlock resolution, tie-breaking, or revert to best-effort service). This MUST be specified for individual negotiation objectives.

2.10.5. Experimental and Example Objective Options

The names "EX0" through "EX9" have been reserved for experimental options. Multiple names have been assigned because a single experiment may use multiple options simultaneously. These experimental options are highly likely to have different meanings when used for different experiments. Therefore, they SHOULD NOT be used without an explicit human decision and MUST NOT be used in unmanaged networks such as home networks.

These names are also RECOMMENDED for use in documentation examples.

3. Implementation Status [RFC Editor: please remove]

Two prototype implementations of GRASP have been made.

3.1. BUPT C++ Implementation

- o Name: BaseNegotiator.cpp, msg.cpp, Client.cpp, Server.cpp
- o Description: C++ implementation of GRASP core and API
- o Maturity: Prototype code, interoperable between Ubuntu.
- o Coverage: Corresponds to draft-carpenter-anima-gdn-protocol-03. Since it was implemented based on the old version draft, the most significant limitations comparing to current protocol design include:
  - * Not support CBOR
  - * Not support Flooding
  - * Not support loop avoidance
  - * only coded for IPv6, any IPv4 is accidental
- o Licensing: Huawei License.
- o Experience: https://github.com/liubingpang/IETF-Anima-Signaling-Protocol/blob/master/README.md
- o Contact: https://github.com/liubingpang/IETF-Anima-Signaling-Protocol

3.2. Python Implementation

- Name: graspy
- Description: Python 3 implementation of GRASP core and API.
- Maturity: Prototype code, interoperable between Windows 7 and Linux.
- Coverage: Corresponds to draft-ietf-anima-grasp-13. Limitations include:
  * insecure: uses a dummy ACP module
  * only coded for IPv6, any IPv4 is accidental
  * FQDN and URI locators incompletely supported
  * no code for rapid mode
  * relay code is lazy (no rate control)
  * all unicast transactions use TCP (no unicast UDP). Experimental code for unicast UDP proved to be complex and brittle.
  * optional Objective option in Response messages not implemented
  * workarounds for defects in Python socket module and Windows socket peculiarities
- Licensing: Simplified BSD
- Experience: Tested on Windows, Linux and MacOS. https://www.cs.auckland.ac.nz/~brian/graspy/graspy.pdf
- Contact: https://www.cs.auckland.ac.nz/~brian/graspy/

4. Security Considerations

A successful attack on negotiation-enabled nodes would be extremely harmful, as such nodes might end up with a completely undesirable configuration that would also adversely affect their peers. GRASP nodes and messages therefore require full protection. As explained in Section 2.5.1, GRASP MUST run within a secure environment such as the Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane], except for the constrained instances described in Section 2.5.2.

- Authentication

  A cryptographically authenticated identity for each device is needed in an autonomic network. It is not safe to assume that a large network is physically secured against interference or that all personnel are trustworthy. Each autonomic node MUST be capable of proving its identity and authenticating its messages. GRASP relies on a separate external certificate-based security mechanism to support authentication, data integrity protection, and anti-replay protection.

  Since GRASP must be deployed in an existing secure environment, the protocol itself specifies nothing concerning the trust anchor and certification authority. For example, in the Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane], all nodes can trust each other and the ASAs installed in them.

  If GRASP is used temporarily without an external security mechanism, for example during system bootstrap (Section 2.5.1), the Session ID (Section 2.7) will act as a nonce to provide limited protection against third parties injecting responses. A full analysis of the secure bootstrap process is in [I-D.ietf-anima-bootstrapping-keyinfra].

- Authorization and Roles

  The GRASP protocol is agnostic about the roles and capabilities of individual ASAs and about which objectives a particular ASA is authorized to support. An implementation might support precautions such as allowing only one ASA in a given node to modify a given objective, but this may not be appropriate in all cases. For example, it might be operationally useful to allow an old and a new version of the same ASA to run simultaneously during an overlap period. These questions are out of scope for the present specification.

- Privacy and confidentiality

  GRASP is intended for network management purposes involving network elements, not end hosts. Therefore, no personal information is expected to be involved in the signaling protocol, so there should be no direct impact on personal privacy. Nevertheless, applications that do convey personal information cannot be excluded. Also, traffic flow paths, VPNs, etc. could be negotiated, which could be of interest for traffic analysis. Operators generally want to conceal details of their network topology and traffic density from outsiders. Therefore, since insider attacks cannot be excluded in a large network, the security mechanism for the protocol MUST provide message confidentiality. This is why Section 2.5.1 requires either an ACP or an alternative security mechanism.

- Link-local multicast security

  GRASP has no reasonable alternative to using link-local multicast for Discovery or Flood Synchronization messages and these messages are sent in clear and with no authentication. They are only sent on interfaces within the autonomic network (see Section 2.1 and Section 2.5.1). They are however available to on-link eavesdroppers, and could be forged by on-link attackers. In the case of Discovery, the Discovery Responses are unicast and will therefore be protected (Section 2.5.1), and an untrusted forger will not be able to receive responses. In the case of Flood Synchronization, an on-link eavesdropper will be able to receive the flooded objectives but there is no response message to consider. Some precautions for Flood Synchronization messages are suggested in Section 2.5.6.2.

- DoS Attack Protection

  GRASP discovery partly relies on insecure link-local multicast. Since routers participating in GRASP sometimes relay discovery messages from one link to another, this could be a vector for denial of service attacks. Some mitigations are specified in Section 2.5.4. However, malicious code installed inside the Autonomic Control Plane could always launch DoS attacks consisting of spurious discovery messages, or of spurious discovery responses. It is important that firewalls prevent any GRASP messages from entering the domain from an unknown source.

- Security during bootstrap and discovery

  A node cannot trust GRASP traffic from other nodes until the security environment (such as the ACP) has identified the trust anchor and can authenticate traffic by validating certificates for other nodes. Also, until it has succesfully enrolled [I-D.ietf-anima-bootstrapping-keyinfra] a node cannot assume that other nodes are able to authenticate its own traffic. Therefore, GRASP discovery during the bootstrap phase for a new device will inevitably be insecure. Secure synchronization and negotiation will be impossible until enrollment is complete. Further details are given in Section 2.5.2.

- Security of discovered locators

When GRASP discovery returns an IP address, it MUST be that of a node within the secure environment (Section 2.5.1). If it returns an FQDN or a URI, the ASA that receives it MUST NOT assume that the target of the locator is within the secure environment.

5. CDDL Specification of GRASP

```
<CODE BEGINS>
grasp-message = (message .within message-structure) / noop-message

message-structure = [MESSAGE_TYPE, session-id, ?initiator,
                     *grasp-option]

MESSAGE_TYPE = 0..255
session-id = 0..4294967295 ;up to 32 bits
grasp-option = any

message /= discovery-message
discovery-message = [M_DISCOVERY, session-id, initiator, objective]

message /= response-message ;response to Discovery
response-message = [M_RESPONSE, session-id, initiator, ttl,
                   (+locator-option // divert-option), ?objective]

message /= synch-message ;response to Synchronization request
synch-message = [M_SYNCH, session-id, objective]

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

message /= request-negotiation-message
request-negotiation-message = [M_REQ_NEG, session-id, objective]

message /= request-synchronization-message
request-synchronization-message = [M_REQ_SYN, session-id, objective]

message /= negotiation-message
negotiation-message = [M_NEGOTIATE, session-id, objective]

message /= end-message
end-message = [M_END, session-id, accept-option / decline-option ]

message /= wait-message
wait-message = [M_WAIT, session-id, waiting-time]

message /= invalid-message
invalid-message = [M_INVALID, session-id, ?any]
```

```
noop-message = [M_NOOP]

divert-option = [O_DIVERT, +locator-option]

accept-option = [O_ACCEPT]

decline-option = [O_DECLINE, ?reason]
reason = text  ;optional UTF-8 error message

waiting-time = 0..4294967295 ; in milliseconds
ttl = 0..4294967295 ; in milliseconds

locator-option /= [O_IPv4_LOCATOR, ipv4-address,
                   transport-proto, port-number]
ipv4-address = bytes .size 4

locator-option /= [O_IPv6_LOCATOR, ipv6-address,
                   transport-proto, port-number]
ipv6-address = bytes .size 16

locator-option /= [O_FQDN_LOCATOR, text, transport-proto, port-number]

locator-option /= [O_URI_LOCATOR, text,
                   transport-proto / null, port-number / null]

transport-proto = IPPROTO_TCP / IPPROTO_UDP
IPPROTO_TCP = 6
IPPROTO_UDP = 17
port-number = 0..65535

initiator = ipv4-address / ipv6-address

objective-flags = uint .bits objective-flag

objective-flag = &(
  F_DISC: 0    ; valid for discovery
  F_NEG: 1     ; valid for negotiation
  F_SYNCH: 2   ; valid for synchronization
  F_NEG_DRY: 3 ; negotiation is dry-run
)

objective = [objective-name, objective-flags, loop-count, ?objective-value]

objective-name = text ;see section "Format of Objective Options"

objective-value = any

loop-count = 0..255
```

```
; Constants for message types and option types

M_NOOP = 0
M_DISCOVERY = 1
M_RESPONSE = 2
M_REQ_NEG = 3
M_REQ_SYN = 4
M_NEGOTIATE = 5
M_END = 6
M_WAIT = 7
M_SYNCH = 8
M_FLOOD = 9
M_INVALID = 99

O_DIVERT = 100
O_ACCEPT = 101
O_DECLINE = 102
O_IPv6_LOCATOR = 103
O_IPv4_LOCATOR = 104
O_FQDN_LOCATOR = 105
O_URI_LOCATOR = 106
<CODE ENDS>
```

6. IANA Considerations

This document defines the GeneRic Autonomic Signaling Protocol (GRASP).

Section 2.6 explains the following link-local multicast addresses, which IANA is requested to assign for use by GRASP:

ALL_GRASP_NEIGHBORS multicast address (IPv6): (TBD1). Assigned in the IPv6 Link-Local Scope Multicast Addresses registry.

ALL_GRASP_NEIGHBORS multicast address (IPv4): (TBD2). Assigned in the IPv4 Multicast Local Network Control Block.

Section 2.6 explains the following User Port, which IANA is requested to assign for use by GRASP for both UDP and TCP:

GRASP_LISTEN_PORT: (TBD3)
Service Name: Generic Autonomic Signaling Protocol (GRASP)
Transport Protocols: UDP, TCP
Assignee: iesg@ietf.org
Contact: chair@ietf.org
Description: See Section 2.6
Reference: RFC XXXX (this document)

The IANA is requested to create a GRASP Parameter Registry including two registry tables. These are the GRASP Messages and Options Table and the GRASP Objective Names Table.

GRASP Messages and Options Table. The values in this table are names paired with decimal integers. Future values MUST be assigned using the Standards Action policy defined by [RFC8126]. The following initial values are assigned by this document:

M_NOOP = 0
M_DISCOVERY = 1
M_RESPONSE = 2
M_REQ_NEG = 3
M_REQ_SYN = 4
M_NEGOTIATE = 5
M_END = 6
M_WAIT = 7
M_SYNCH = 8
M_FLOOD = 9
M_INVALID = 99

O_DIVERT = 100
O_ACCEPT = 101
O_DECLINE = 102
O_IPv6_LOCATOR = 103
O_IPv4_LOCATOR = 104
O_FQDN_LOCATOR = 105
O_URI_LOCATOR = 106

GRASP Objective Names Table. The values in this table are UTF-8 strings which MUST NOT include a colon (":"), according to Section 2.10.1. Future values MUST be assigned using the Specification Required policy defined by [RFC8126].

To assist expert review of a new objective, the specification should include a precise description of the format of the new objective, with sufficient explanation of its semantics to allow independent implementations. See Section 2.10.3 for more details. If the new objective is similar in name or purpose to a previously registered objective, the specification should explain why a new objective is justified.

The following initial values are assigned by this document:

EX0
EX1
EX2
EX3
EX4
EX5
EX6
EX7
EX8
EX9

7. Acknowledgements

A major contribution to the original version of this document was made by Sheng Jiang and significant contributions were made by Toerless Eckert. Significant early review inputs were received from Joel Halpern, Barry Leiba, Charles E. Perkins, and Michael Richardson. William Atwood provided important assistance in debugging a prototype implementation.

Valuable comments were received from Michael Behringer, Jeferson Campos Nobre, Laurent Ciavaglia, Zongpeng Du, Yu Fu, Joel Jaeggli, Zhenbin Li, Dimitri Papadimitriou, Pierre Peloso, Reshad Rahman, Markus Stenberg, Martin Stiemerling, Rene Struik, Martin Thomson, Dacheng Zhang, and participants in the NMRG research group, the ANIMA working group, and the IESG.

8. References

8.1. Normative References

[I-D.greevenbosch-appsawg-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR data structures", draft-greevenbosch-appsawg-cbor-cddl-11 (work in progress), July 2017.

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-07 (work in progress), July 2017.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

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

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

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <http://www.rfc-editor.org/info/rfc8085>.

8.2. Informative References

[I-D.carpenter-anima-asa-guidelines]
Carpenter, B. and S. Jiang, "Guidelines for Autonomic Service Agents", draft-carpenter-anima-asa-guidelines-02 (work in progress), July 2017.

[I-D.chaparadza-intarea-igcp]
Behringer, M., Chaparadza, R., Petre, R., Li, X., and H. Mahkonen, "IP based Generic Control Protocol (IGCP)", draft-chaparadza-intarea-igcp-00 (work in progress), July 2011.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-07 (work in progress), July 2017.

[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-04 (work in progress), July 2017.

[I-D.ietf-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-ietf-anima-stable-connectivity-03 (work in progress), July 2017.

[I-D.liu-anima-grasp-api]
Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", draft-liu-anima-grasp-api-04 (work in progress), June 2017.

[I-D.stenberg-anima-adncp]
Stenberg, M., "Autonomic Distributed Node Consensus Protocol", draft-stenberg-anima-adncp-00 (work in progress), March 2015.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <http://www.rfc-editor.org/info/rfc2205>.

[RFC2334] Luciani, J., Armitage, G., Halpern, J., and N. Doraswamy, "Server Cache Synchronization Protocol (SCSP)", RFC 2334, DOI 10.17487/RFC2334, April 1998, <http://www.rfc-editor.org/info/rfc2334>.

[RFC2608] Guttman, E., Perkins, C., Veizades, J., and M. Day, "Service Location Protocol, Version 2", RFC 2608, DOI 10.17487/RFC2608, June 1999, <http://www.rfc-editor.org/info/rfc2608>.

[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, <http://www.rfc-editor.org/info/rfc2865>.

[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, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002, <http://www.rfc-editor.org/info/rfc3416>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <http://www.rfc-editor.org/info/rfc3493>.

[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, <http://www.rfc-editor.org/info/rfc4861>.

[RFC5612] Eronen, P. and D. Harrington, "Enterprise Number for Documentation Use", RFC 5612, DOI 10.17487/RFC5612, August 2009, <http://www.rfc-editor.org/info/rfc5612>.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, DOI 10.17487/RFC5971, October 2010, <http://www.rfc-editor.org/info/rfc5971>.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <http://www.rfc-editor.org/info/rfc6206>.

[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, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

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

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

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

[RFC7558] Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions", RFC 7558, DOI 10.17487/RFC7558, July 2015, <http://www.rfc-editor.org/info/rfc7558>.

[RFC7564] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 7564, DOI 10.17487/RFC7564, May 2015, <http://www.rfc-editor.org/info/rfc7564>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7576>.

[RFC7787] Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", RFC 7787, DOI 10.17487/RFC7787, April 2016, <http://www.rfc-editor.org/info/rfc7787>.

[RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <http://www.rfc-editor.org/info/rfc7788>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

[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, <http://www.rfc-editor.org/info/rfc8126>.

Appendix A. Open Issues [RFC Editor: This section should be empty. Please remove]

- o 68. (Placeholder)

Appendix B. Closed Issues [RFC Editor: Please remove]

- o 1. UDP vs TCP: For now, this specification suggests UDP and TCP as message transport mechanisms. This is not clarified yet. UDP is good for short conversations, is necessary for multicast discovery, and generally fits the discovery and divert scenarios well. However, it will cause problems with large messages. TCP is good for stable and long sessions, with a little bit of time consumption during the session establishment stage. If messages exceed a reasonable MTU, a TCP mode will be required in any case. This question may be affected by the security discussion.

  RESOLVED by specifying UDP for short message and TCP for longer one.

- o 2. DTLS or TLS vs built-in security mechanism. For now, this specification has chosen a PKI based built-in security mechanism based on asymmetric cryptography. However, (D)TLS might be chosen as security solution to avoid duplication of effort. It also allows essentially similar security for short messages over UDP and longer ones over TCP. The implementation trade-offs are different. The current approach requires expensive asymmetric cryptographic calculations for every message. (D)TLS has startup overheads but cheaper crypto per message. DTLS is less mature than TLS.

  RESOLVED by specifying external security (ACP or (D)TLS).

- o The following open issues applied only if the original security model was retained:

  * 2.1. For replay protection, GRASP currently requires every participant to have an NTP-synchronized clock. Is this OK for low-end devices, and how does it work during device bootstrapping? We could take the Timestamp out of signature option, to become an independent and OPTIONAL (or RECOMMENDED) option.

  * 2.2. The Signature Option states that this option could be any place in a message. Wouldn’t it be better to specify a position (such as the end)? That would be much simpler to implement.

RESOLVED by changing security model.

o 3. DoS Attack Protection needs work.

RESOLVED by adding text.

o 4. Should we consider preferring a text-based approach to discovery (after the initial discovery needed for bootstrapping)? This could be a complementary mechanism for multicast based discovery, especially for a very large autonomic network. Centralized registration could be automatically deployed incrementally. At the very first stage, the repository could be empty; then it could be filled in by the objectives discovered by different devices (for example using Dynamic DNS Update). The more records are stored in the repository, the less the multicast-based discovery is needed. However, if we adopt such a mechanism, there would be challenges: stateful solution, and security.

RESOLVED for now by adding optional use of DNS-SD by ASAs. Subsequently removed by editors as irrelevant to GRASP istelf.

o 5. Need to expand description of the minimum requirements for the specification of an individual discovery, synchronization or negotiation objective.

RESOLVED for now by extra wording.

o 6. Use case and protocol walkthrough. A description of how a node starts up, performs discovery, and conducts negotiation and synchronisation for a sample use case would help readers to understand the applicability of this specification. Maybe it should be an artificial use case or maybe a simple real one, based on a conceptual API. However, the authors have not yet decided whether to have a separate document or have it in the protocol document.

RESOLVED: recommend a separate document.

o 7. Cross-check against other ANIMA WG documents for consistency and gaps.

RESOLVED: Satisfied by WGLC.

o 8. Consideration of ADNCP proposal.

RESOLVED by adding optional use of DNCP for flooding-type synchronization.

o 9. Clarify how a GDNP instance knows whether it is running inside the ACP. (Sheng)

RESOLVED by improved text.

o 10. Clarify how a non-ACP GDNP instance initiates (D)TLS. (Sheng)

RESOLVED by improved text and declaring DTLS out of scope for this draft.

o 11. Clarify how UDP/TCP choice is made. (Sheng) [Like DNS? - Brian]

RESOLVED by improved text.

o 12. Justify that IP address within ACP or (D)TLS environment is sufficient to prove AN identity; or explain how Device Identity Option is used. (Sheng)

RESOLVED for now: we assume that all ASAs in a device are trusted as soon as the device is trusted, so they share credentials. In that case the Device Identity Option is useless. This needs to be reviewed later.

o 13. Emphasise that negotiation/synchronization are independent from discovery, although the rapid discovery mode includes the first step of a negotiation/synchronization. (Sheng)

RESOLVED by improved text.

o 14. Do we need an unsolicited flooding mechanism for discovery (for discovery results that everyone needs), to reduce scaling impact of flooding discovery messages? (Toerless)

RESOLVED: Yes, added to requirements and solution.

o 15. Do we need flag bits in Objective Options to distinguish distinguish Synchronization and Negotiation "Request" or rapid mode "Discovery" messages? (Bing)

RESOLVED: yes, work on the API showed that these flags are essential.

o 16. (Related to issue 14). Should we revive the "unsolicited Response" for flooding synchronisation data? This has to be done carefully due to the well-known issues with flooding, but it could be useful, e.g. for Intent distribution, where DNCP doesn't seem applicable.

RESOLVED: Yes, see #14.

- 17. Ensure that the discovery mechanism is completely proof against loops and protected against duplicate responses.

  RESOLVED: Added loop count mechanism.

- 18. Discuss the handling of multiple valid discovery responses.

  RESOLVED: Stated that the choice must be available to the ASA but GRASP implementation should pick a default.

- 19. Should we use a text-oriented format such as JSON/CBOR instead of native binary TLV format?

  RESOLVED: Yes, changed to CBOR.

- 20. Is the Divert option needed? If a discovery response provides a valid IP address or FQDN, the recipient doesn't gain any extra knowledge from the Divert. On the other hand, the presence of Divert informs the receiver that the target is off-link, which might be useful sometimes.

  RESOLVED: Decided to keep Divert option.

- 21. Rename the protocol as GRASP (GeneRic Autonomic Signaling Protocol)?

  RESOLVED: Yes, name changed.

- 22. Does discovery mechanism scale robustly as needed? Need hop limit on relaying?

  RESOLVED: Added hop limit.

- 23. Need more details on TTL for caching discovery responses.

  RESOLVED: Done.

- 24. Do we need "fast withdrawal" of discovery responses?

  RESOLVED: This doesn't seem necessary. If an ASA exits or stops supporting a given objective, peers will fail to start future sessions and will simply repeat discovery.

- 25. Does GDNP discovery meet the needs of multi-hop DNS-SD?

  RESOLVED: Decided not to consider this further as a GRASP protocol issue. GRASP objectives could embed DNS-SD formats if needed.

- 26. Add a URL type to the locator options (for security bootstrap etc.)

  RESOLVED: Done, later renamed as URI.

- 27. Security of Flood multicasts (Section 2.5.6.2).

  RESOLVED: added text.

- 28. Does ACP support secure link-local multicast?

  RESOLVED by new text in the Security Considerations.

- 29. PEN is used to distinguish vendor options. Would it be better to use a domain name? Anything unique will do.

  RESOLVED: Simplified this by removing PEN field and changing naming rules for objectives.

- 30. Does response to discovery require randomized delays to mitigate amplification attacks?

  RESOLVED: WG feedback is that it’s unnecessary.

- 31. We have specified repeats for failed discovery etc. Is that sufficient to deal with sleeping nodes?

  RESOLVED: WG feedback is that it’s unnecessary to say more.

- 32. We have one-to-one synchronization and flooding synchronization. Do we also need selective flooding to a subset of nodes?

  RESOLVED: This will be discussed as a protocol extension in a separate draft (draft-liu-anima-grasp-distribution).

- 33. Clarify if/when discovery needs to be repeated.

  RESOLVED: Done.

- 34. Clarify what is mandatory for running in ACP, expand discussion of security boundary when running with no ACP - might rely on the local PKI infrastructure.

RESOLVED: Done.

- 35. State that role-based authorization of ASAs is out of scope for GRASP. GRASP doesn't recognize/handle any "roles".

  RESOLVED: Done.

- 36. Reconsider CBOR definition for PEN syntax. (objective-name = text / [pen, text] ; pen = uint)

  RESOLVED: See issue 29.

- 37. Are URI locators really needed?

  RESOLVED: Yes, e.g. for security bootstrap discovery, but added note that addresses are the normal case (same for FQDN locators).

- 38. Is Session ID sufficient to identify relayed responses? Isn't the originator's address needed too?

  RESOLVED: Yes, this is needed for multicast messages and their responses.

- 39. Clarify that a node will contain one GRASP instance supporting multiple ASAs.

  RESOLVED: Done.

- 40. Add a "reason" code to the DECLINE option?

  RESOLVED: Done.

- 41. What happens if an ASA cannot conveniently use one of the GRASP mechanisms? Do we (a) add a message type to GRASP, or (b) simply pass the discovery results to the ASA so that it can open its own socket?

  RESOLVED: Both would be possible, but (b) is preferred.

- 42. Do we need a feature whereby an ASA can bypass the ACP and use the data plane for efficiency/throughput? This would require discovery to return non-ACP addresses and would evade ACP security.

  RESOLVED: This is considered out of scope for GRASP, but a comment has been added in security considerations.

o 43. Rapid mode synchronization and negotiation is currently limited to a single objective for simplicity of design and implementation. A future consideration is to allow multiple objectives in rapid mode for greater efficiency.

RESOLVED: This is considered out of scope for this version.

o 44. In requirement T9, the words that encryption "may not be required in all deployments" were removed. Is that OK?.

RESOLVED: No objections.

o 45. Device Identity Option is unused. Can we remove it completely?.

RESOLVED: No objections. Done.

o 46. The 'initiator' field in DISCOVER, RESPONSE and FLOOD messages is intended to assist in loop prevention. However, we also have the loop count for that. Also, if we create a new Session ID each time a DISCOVER or FLOOD is relayed, that ID can be disambiguated by recipients. It would be simpler to remove the initiator from the messages, making parsing more uniform. Is that OK?

RESOLVED: Yes. Done.

o 47. REQUEST is a dual purpose message (request negotiation or request synchronization). Would it be better to split this into two different messages (and adjust various message names accordingly)?

RESOLVED: Yes. Done.

o 48. Should the Appendix "Capability Analysis of Current Protocols" be deleted before RFC publication?

RESOLVED: No (per WG meeting at IETF 96).

o 49. Section 2.5.1 Should say more about signaling between two autonomic networks/domains.

RESOLVED: Description of separate GRASP instance added.

o 50. Is Rapid mode limited to on-link only? What happens if first discovery responder does not support Rapid Mode? Section 2.5.5, Section 2.5.6)

RESOLVED: Not limited to on-link. First responder wins.

- 51. Should flooded objectives have a time-to-live before they are deleted from the flood cache? And should they be tagged in the cache with their source locator?

  RESOLVED: TTL added to Flood (and Discovery Response) messages. Cached flooded objectives must be tagged with their originating ASA locator, and multiple copies must be kept if necessary.

- 52. Describe in detail what is allowed and disallowed in an insecure instance of GRASP.

  RESOLVED: Done.

- 53. Tune IANA Considerations to support early assignment request.

- 54. Is there a highly unlikely race condition if two peers simultaneously choose the same Session ID and send each other simultaneous M_REQ_NEG messages?

  RESOLVED: Yes. Enhanced text on Session ID generation, and added precaution when receiving a Request message.

- 55. Could discovery be performed over TCP?

  RESOLVED: Unicast discovery added as an option.

- 56. Change Session-ID to 32 bits?

  RESOLVED: Done.

- 57. Add M_INVALID message?

  RESOLVED: Done.

- 58. Maximum message size?

  RESOLVED by specifying default maximum message size (2048 bytes).

- 59. Add F_NEG_DRY flag to specify a "dry run" objective?.

  RESOLVED: Done.

- 60. Change M_FLOOD syntax to associate a locator with each objective?

RESOLVED: Done.

- 61. Is the SONN constrained instance really needed?

  RESOLVED: Retained but only as an option.

- 62. Is it helpful to tag descriptive text with message names (M_DISCOVER etc.)?

  RESOLVED: Yes, done in various parts of the text.

- 63. Should encryption be MUST instead of SHOULD in Section 2.5.1 and Section 2.5.1?

  RESOLVED: Yes, MUST implement in both cases.

- 64. Should more security text be moved from the main text into the Security Considerations?

  RESOLVED: No, on AD advice.

- 65. Do we need to formally restrict Unicode characters allowed in objective names?

  RESOLVED: No, but need to point to guidance from PRECIS WG.

- 66. Split requirements into separate document?

  RESOLVED: No, on AD advice.

- 67. Remove normative dependency on draft-greevenbosch-appsawg-cbor-cddl?

  RESOLVED: No, on AD advice. In worst case, fix at AUTH48.

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

draft-ietf-anima-grasp-15, 2017-07-07:

Updates following additional IESG comments:

Security (Eric Rescorla): missing brittleness of group security concept, attack via compromised member.

TSV (Mirja Kuehlewind): clarification on the use of UDP, TCP, mandate use of TCP (or other reliable transport).

Clarified that in ACP, UDP is not used at all.

Clarified that GRASP itself needs TCP listen port (was previously written as if this was optional).

draft-ietf-anima-grasp-14, 2017-07-02:

Updates following additional IESG comments:

Updated 2.5.1 and 2.5.2 based on IESG security feedback (specify dependency against security substrate).

Strengthened requirement for reliable transport protocol.

draft-ietf-anima-grasp-13, 2017-06-06:

Updates following additional IESG comments:

Removed all mention of TLS, including SONN, since it was under-specified.

Clarified other text about trust and security model.

Banned Rapid Mode when multicast is insecure.

Explained use of M_INVALID to support extensibility

Corrected details on discovery cache TTL and discovery timeout.

Improved description of multicast UDP w.r.t. RFC8085.

Clarified when transport connections are opened or closed.

Noted that IPPROTO values come from the Protocol Numbers registry

Protocol change: Added protocol and port numbers to URI locator.

Removed inaccurate text about routing protocols

Moved Requirements section to an Appendix.

Other editorial and technical clarifications.

draft-ietf-anima-grasp-12, 2017-05-19:

Updates following IESG comments:

Clarified that GRASP runs in a single addressing realm

Improved wording about FQDN resolution, clarified that URI usage is out of scope.

Clarified description of negotiation timeout.

Noted that ’dry run’ semantics are ASA-dependent

Made the ACP a normative reference

Clarified that LL multicasts are limited to GRASP interfaces

Unicast UDP moved out of scope

Editorial clarifications

draft-ietf-anima-grasp-11, 2017-03-30:

Updates following IETF 98 discussion:

Encryption changed to a MUST implement.

Pointed to guidance on UTF-8 names.

draft-ietf-anima-grasp-10, 2017-03-10:

Updates following IETF Last call:

Protocol change: Specify that an objective with no initial value should have its value field set to CBOR ’null’.

Protocol change: Specify behavior on receiving unrecognized message type.

Noted that UTF-8 names are matched byte-for-byte.

Added brief guidance for Expert Reviewer of new generic objectives.

Numerous editorial improvements and clarifications and minor text rearrangements, none intended to change the meaning.

draft-ietf-anima-grasp-09, 2016-12-15:

Protocol change: Add F_NEG_DRY flag to specify a "dry run" objective.

Protocol change: Change M_FLOOD syntax to associate a locator with each objective.

Concentrated mentions of TLS in one section, with all details out of scope.

Clarified text around constrained instances of GRASP.

Strengthened text restricting LL addresses in locator options.

Clarified description of rapid mode processsing.

Specified that cached discovery results should not be returned on the same interface where they were learned.

Shortened text in "High Level Design Choices"

Dropped the word ’kernel’ to avoid confusion with o/s kernel mode.

Editorial improvements and clarifications.

draft-ietf-anima-grasp-08, 2016-10-30:

Protocol change: Added M_INVALID message.

Protocol change: Increased Session ID space to 32 bits.

Enhanced rules to avoid Session ID clashes.

Corrected and completed description of timeouts for Request messages.

Improved wording about exponential backoff and DoS.

Clarified that discovery relaying is not done by limited security instances.

Corrected and expanded explanation of port used for Discovery Response.

Noted that Discovery message could be sent unicast in special cases.

Added paragraph on extensibility.

Specified default maximum message size.

Added Appendix for sample messages.

Added short protocol overview.

Editorial fixes, including minor re-ordering for readability.

draft-ietf-anima-grasp-07, 2016-09-13:

Protocol change: Added TTL field to Flood message (issue 51).

Protocol change: Added Locator option to Flood message (issue 51).

Protocol change: Added TTL field to Discovery Response message (corrollary to issue 51).

Clarified details of rapid mode (issues 43 and 50).

Description of inter-domain GRASP instance added (issue 49).

Description of limited security GRASP instances added (issue 52).

Strengthened advice to use TCP rather than UDP.

Updated IANA considerations and text about well-known port usage (issue 53).

Amended text about ASA authorization and roles to allow for overlapping ASAs.

Added text recommending that Flood should be repeated periodically.

Editorial fixes.

draft-ietf-anima-grasp-06, 2016-06-27:

Added text on discovery cache timeouts.

Noted that ASAs that are only initiators do not need to respond to discovery message.

Added text on unexpected address changes.

Added text on robust implementation.

Clarifications and editorial fixes for numerous review comments

Added open issues for some review comments.

draft-ietf-anima-grasp-05, 2016-05-13:

Noted in requirement T1 that it should be possible to implement ASAs independently as user space programs.

Protocol change: Added protocol number and port to discovery response. Updated protocol description, CDDL and IANA considerations accordingly.

Clarified that discovery and flood multicasts are handled by the GRASP core, not directly by ASAs.

Clarified that a node may discover an objective without supporting it for synchronization or negotiation.

Added Implementation Status section.

Added reference to SCSP.

Editorial fixes.

draft-ietf-anima-grasp-04, 2016-03-11:

Protocol change: Restored initiator field in certain messages and adjusted relaying rules to provide complete loop detection.

Updated IANA Considerations.

draft-ietf-anima-grasp-03, 2016-02-24:

Protocol change: Removed initiator field from certain messages and adjusted relaying requirement to simplify loop detection. Also clarified narrative explanation of discovery relaying.

Protocol change: Split Request message into two (Request Negotiation and Request Synchronization) and updated other message names for clarity.

Protocol change: Dropped unused Device ID option.

Further clarified text on transport layer usage.

New text about multicast insecurity in Security Considerations.

Various other clarifications and editorial fixes, including moving some material to Appendix.

draft-ietf-anima-grasp-02, 2016-01-13:

Resolved numerous issues according to WG discussions.

Renumbered requirements, added D9.

Protocol change: only allow one objective in rapid mode.

Protocol change: added optional error string to DECLINE option.

Protocol change: removed statement that seemed to say that a Request not preceded by a Discovery should cause a Discovery response. That made no sense, because there is no way the initiator would know where to send the Request.

Protocol change: Removed PEN option from vendor objectives, changed naming rule accordingly.

Protocol change: Added FLOOD message to simplify coding.

Protocol change: Added SYNCH message to simplify coding.

Protocol change: Added initiator id to DISCOVER, RESPONSE and FLOOD messages. But also allowed the relay process for DISCOVER and FLOOD to regenerate a Session ID.

Protocol change: Require that discovered addresses must be global (except during bootstrap).

Protocol change: Receiver of REQUEST message must close socket if no ASA is listening for the objective.

Protocol change: Simplified Waiting message.

Protocol change: Added No Operation message.

Renamed URL locator type as URI locator type.

Updated CDDL definition.

Various other clarifications and editorial fixes.

draft-ietf-anima-grasp-01, 2015-10-09:

Updated requirements after list discussion.

Changed from TLV to CBOR format - many detailed changes, added co-author.

Tightened up loop count and timeouts for various cases.

Noted that GRASP does not provide transactional integrity.

Various other clarifications and editorial fixes.

draft-ietf-anima-grasp-00, 2015-08-14:

File name and protocol name changed following WG adoption.

Added URL locator type.

draft-carpenter-anima-gdn-protocol-04, 2015-06-21:

Tuned wording around hierarchical structure.

Changed "device" to "ASA" in many places.

Reformulated requirements to be clear that the ASA is the main customer for signaling.

Added requirement for flooding unsolicited synch, and added it to protocol spec. Recognized DNCP as alternative for flooding synch data.

Requirements clarified, expanded and rearranged following design team discussion.

Clarified that GDNP discovery must not be a prerequisite for GDNP negotiation or synchronization (resolved issue 13).

Specified flag bits for objective options (resolved issue 15).

Clarified usage of ACP vs TLS/DTLS and TCP vs UDP (resolved issues 9,10,11).

Updated DNCP description from latest DNCP draft.

Editorial improvements.

draft-carpenter-anima-gdn-protocol-03, 2015-04-20:

Removed intrinsic security, required external security

Format changes to allow DNCP co-existence

Recognized DNS-SD as alternative discovery method.

Editorial improvements

draft-carpenter-anima-gdn-protocol-02, 2015-02-19:

Tuned requirements to clarify scope,

Clarified relationship between types of objective,

Clarified that objectives may be simple values or complex data structures,

Improved description of objective options,

Added loop-avoidance mechanisms (loop count and default timeout, limitations on discovery relaying and on unsolicited responses),

Allow multiple discovery objectives in one response,

Provided for missing or multiple discovery responses,

Indicated how modes such as "dry run" should be supported,

Minor editorial and technical corrections and clarifications,

Reorganized future work list.

draft-carpenter-anima-gdn-protocol-01, restructured the logical flow of the document, updated to describe synchronization completely, add unsolicited responses, numerous corrections and clarifications, expanded future work list, 2015-01-06.

draft-carpenter-anima-gdn-protocol-00, combination of draft-jiang-config-negotiation-ps-03 and draft-jiang-config-negotiation-protocol-02, 2014-10-08.

Appendix D. Example Message Formats

For readers unfamiliar with CBOR, this appendix shows a number of example GRASP messages conforming to the CDDL syntax given in Section 5. Each message is shown three times in the following formats:

1. CBOR diagnostic notation.

2. Similar, but showing the names of the constants. (Details of the flag bit encoding are omitted.)

3. Hexadecimal version of the CBOR wire format.

Long lines are split for display purposes only.

D.1. Discovery Example

The initiator (2001:db8:f000:baaa:28cc:dc4c:9703:6781) multicasts a discovery message looking for objective EX1:

```
[1, 13948744, h'20010db8f000baaa28ccdc4c97036781', ["EX1", 5, 2, 0]]
[M_DISCOVERY, 13948744, h'20010db8f000baaa28ccdc4c97036781',
              ["EX1", F_SYNCH_bits, 2, 0]]
h'84011a00d4d7485020010db8f000baaa28ccdc4c970367818463455831050200'
```

A peer (2001:0db8:f000:baaa:f000:baaa:f000:baaa) responds with a locator:

```
[2, 13948744, h'20010db8f000baaa28ccdc4c97036781', 60000,
              [103, h'20010db8f000baaaf000baaaf000baaa', 6, 49443]]
[M_RESPONSE, 13948744, h'20010db8f000baaa28ccdc4c97036781', 60000,
              [O_IPv6_LOCATOR, h'20010db8f000baaaf000baaaf000baaa',
               IPPROTO_TCP, 49443]]
h'85021a00d4d7485020010db8f000baaa28ccdc4c9703678119ea6084186750
  20010db8f000baaaf000baaaf000baaa0619c123'
```

D.2. Flood Example

The initiator multicasts a flood message. The single objective has a null locator. There is no response:

```
[9, 3504974, h'20010db8f000baaa28ccdc4c97036781', 10000,
            [["EX1", 5, 2, ["Example 1 value=", 100]],[] ] ]
[M_FLOOD, 3504974, h'20010db8f000baaa28ccdc4c97036781', 10000,
            [["EX1", F_SYNCH_bits, 2, ["Example 1 value=", 100]],[] ] ]
h'86091a00357b4e5020010db8f000baaa28ccdc4c97036781192710
  828463455831050282704578616d706c6520312076616c75653d186480'
```

D.3. Synchronization Example

Following successful discovery of objective EX2, the initiator unicasts a request:

```
[4, 4038926, ["EX2", 5, 5, 0]]
[M_REQ_SYN, 4038926, ["EX2", F_SYNCH_bits, 5, 0]]
h'83041a003da10e8463455832050500'
```

The peer responds with a value:

```
[8, 4038926, ["EX2", 5, 5, ["Example 2 value=", 200]]]
[M_SYNCH, 4038926, ["EX2", F_SYNCH_bits, 5, ["Example 2 value=", 200]]]
h'83081a003da10e8463455832050582704578616d706c6520322076616c75653d18c8'
```

D.4. Simple Negotiation Example

Following successful discovery of objective EX3, the initiator unicasts a request:

```
[3, 802813, ["EX3", 3, 6, ["NZD", 47]]]
[M_REQ_NEG, 802813, ["EX3", F_NEG_bits, 6, ["NZD", 47]]]
h'83031a000c3ffd846345583303068263 4e5a44182f'
```

The peer responds with immediate acceptance. Note that no objective is needed, because the initiator's request was accepted without change:

```
[6, 802813, [101]]
[M_END , 802813, [O_ACCEPT]]
h'83061a000c3ffd811865'
```

D.5. Complete Negotiation Example

Again the initiator unicasts a request:

```
[3, 13767778, ["EX3", 3, 6, ["NZD", 410]]]
[M_REQ_NEG, 13767778, ["EX3", F_NEG_bits, 6, ["NZD", 410]]]
h'83031a00d2146284634558330306826 34e5a4419019a'
```

The responder starts to negotiate (making an offer):

```
[5, 13767778, ["EX3", 3, 6, ["NZD", 80]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 6, ["NZD", 80]]]
h'83051a00d21462846345583303068263 4e5a441850'
```

The initiator continues to negotiate (reducing its request, and note that the loop count is decremented):

```
[5, 13767778, ["EX3", 3, 5, ["NZD", 307]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 5, ["NZD", 307]]]
h'83051a00d2146284634558330305826 34e5a44190133'
```

The responder asks for more time:

```
[7, 13767778, 34965]
[M_WAIT, 13767778, 34965]
h'83071a00d21462198895'
```

The responder continues to negotiate (increasing its offer):

```
[5, 13767778, ["EX3", 3, 4, ["NZD", 120]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 4, ["NZD", 120]]]
h'83051a00d214628463455833030482634e5a441878'
```

The initiator continues to negotiate (reducing its request):

```
[5, 13767778, ["EX3", 3, 3, ["NZD", 246]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 3, ["NZD", 246]]]
h'83051a00d214628463455833030382634e5a4418f6'
```

The responder refuses to negotiate further:

```
[6, 13767778, [102, "Insufficient funds"]]
[M_END , 13767778, [O_DECLINE, "Insufficient funds"]]
h'83061a00d2146282186672496e73756666696369656e742066756e6473'
```

This negotiation has failed. If either side had sent [M_END, 13767778, [O_ACCEPT]] it would have succeeded, converging on the objective value in the preceding M_NEGOTIATE. Note that apart from the initial M_REQ_NEG, the process is symmetrical.

Appendix E. Requirement Analysis of Discovery, Synchronization and Negotiation

This section discusses the requirements for discovery, negotiation and synchronization capabilities. The primary user of the protocol is an autonomic service agent (ASA), so the requirements are mainly expressed as the features needed by an ASA. A single physical device might contain several ASAs, and a single ASA might manage several technical objectives. If a technical objective is managed by several ASAs, any necessary coordination is outside the scope of the GRASP signaling protocol. Furthermore, requirements for ASAs themselves, such as the processing of Intent [RFC7575], are out of scope for the present document.

E.1. Requirements for Discovery

D1. ASAs may be designed to manage any type of configurable device or software, as required in Appendix E.2. A basic requirement is therefore that the protocol can represent and discover any kind of technical objective (as defined in Section 2.1) among arbitrary subsets of participating nodes.

In an autonomic network we must assume that when a device starts up it has no information about any peer devices, the network structure, or what specific role it must play. The ASA(s) inside the device are in the same situation. In some cases, when a new application session starts up within a device, the device or ASA may again lack information about relevant peers. For example, it might be necessary to set up resources on multiple other devices, coordinated and matched to each other so that there is no wasted resource. Security settings might also need updating to allow for the new device or user. The relevant peers may be different for different technical objectives. Therefore discovery needs to be repeated as often as necessary to find peers capable of acting as counterparts for each objective that a discovery initiator needs to handle. From this background we derive the next three requirements:

D2. When an ASA first starts up, it may have no knowledge of the specific network to which it is attached. Therefore the discovery process must be able to support any network scenario, assuming only that the device concerned is bootstrapped from factory condition.

D3. When an ASA starts up, it must require no configured location information about any peers in order to discover them.

D4. If an ASA supports multiple technical objectives, relevant peers may be different for different discovery objectives, so discovery needs to be performed separately to find counterparts for each objective. Thus, there must be a mechanism by which an ASA can separately discover peer ASAs for each of the technical objectives that it needs to manage, whenever necessary.

D5. Following discovery, an ASA will normally perform negotiation or synchronization for the corresponding objectives. The design should allow for this by conveniently linking discovery to negotiation and synchronization. It may provide an optional mechanism to combine discovery and negotiation/synchronization in a single protocol exchange.

D6. Some objectives may only be significant on the local link, but others may be significant across the routed network and require off-link operations. Thus, the relevant peers might be immediate neighbors on the same layer 2 link, or they might be more distant and only accessible via layer 3. The mechanism must therefore provide both on-link and off-link discovery of ASAs supporting specific technical objectives.

D7. The discovery process should be flexible enough to allow for special cases, such as the following:

- o During initialization, a device must be able to establish mutual trust with autonomic nodes elsewhere in the network and participate in an authentication mechanism. Although this will inevitably start with a discovery action, it is a special case precisely because trust is not yet established. This topic is the subject of [I-D.ietf-anima-bootstrapping-keyinfra]. We require that once trust has been established for a device, all ASAs within the device inherit the device’s credentials and are also trusted. This does not preclude the device having multiple credentials.

- Depending on the type of network involved, discovery of other central functions might be needed, such as the Network Operations Center (NOC) [I-D.ietf-anima-stable-connectivity]. The protocol must be capable of supporting such discovery during initialization, as well as discovery during ongoing operation.

D8. The discovery process must not generate excessive traffic and must take account of sleeping nodes.

D9. There must be a mechanism for handling stale discovery results.

E.2. Requirements for Synchronization and Negotiation Capability

Autonomic networks need to be able to manage many different types of parameter and consider many dimensions, such as latency, load, unused or limited resources, conflicting resource requests, security settings, power saving, load balancing, etc. Status information and resource metrics need to be shared between nodes for dynamic adjustment of resources and for monitoring purposes. While this might be achieved by existing protocols when they are available, the new protocol needs to be able to support parameter exchange, including mutual synchronization, even when no negotiation as such is required. In general, these parameters do not apply to all participating nodes, but only to a subset.

SN1. A basic requirement for the protocol is therefore the ability to represent, discover, synchronize and negotiate almost any kind of network parameter among selected subsets of participating nodes.

SN2. Negotiation is an iterative request/response process that must be guaranteed to terminate (with success or failure). While tie-breaking rules must be defined specifically for each use case, the protocol should have some general mechanisms in support of loop and deadlock prevention, such as hop count limits or timeouts.

SN3. Synchronization must be possible for groups of nodes ranging from small to very large.

SN4. To avoid "reinventing the wheel", the protocol should be able to encapsulate the data formats used by existing configuration protocols (such as NETCONF/YANG) in cases where that is convenient.

SN5. Human intervention in complex situations is costly and error-prone. Therefore, synchronization or negotiation of parameters without human intervention is desirable whenever the coordination of multiple devices can improve overall network performance. It follows that the protocol's resource requirements must be small enough to fit in any device that would otherwise need human intervention. The issue of running in constrained nodes is discussed in [I-D.ietf-anima-reference-model].

SN6. Human intervention in large networks is often replaced by use of a top-down network management system (NMS). It therefore follows that the protocol, as part of the Autonomic Networking Infrastructure, should be capable of running in any device that would otherwise be managed by an NMS, and that it can co-exist with an NMS, and with protocols such as SNMP and NETCONF.

SN7. Specific autonomic features are expected to be implemented by individual ASAs, but the protocol must be general enough to allow them. Some examples follow:

- o Dependencies and conflicts: In order to decide upon a configuration for a given device, the device may need information from neighbors. This can be established through the negotiation procedure, or through synchronization if that is sufficient. However, a given item in a neighbor may depend on other information from its own neighbors, which may need another negotiation or synchronization procedure to obtain or decide. Therefore, there are potential dependencies and conflicts among negotiation or synchronization procedures. Resolving dependencies and conflicts is a matter for the individual ASAs involved. To allow this, there need to be clear boundaries and convergence mechanisms for negotiations. Also some mechanisms are needed to avoid loop dependencies or uncontrolled growth in a tree of dependencies. It is the ASA designer's responsibility to avoid or detect looping dependencies or excessive growth of dependency trees. The protocol's role is limited to bilateral signaling between ASAs, and the avoidance of loops during bilateral signaling.

- o Recovery from faults and identification of faulty devices should be as automatic as possible. The protocol's role is limited to discovery, synchronization and negotiation. These processes can occur at any time, and an ASA may need to repeat any of these steps when the ASA detects an event such as a negotiation counterpart failing.

- o Since a major goal is to minimize human intervention, it is necessary that the network can in effect "think ahead" before changing its parameters. One aspect of this is an ASA that relies on a knowledge base to predict network behavior. This is out of scope for the signaling protocol. However, another aspect is forecasting the effect of a change by a "dry run" negotiation before actually installing the change. Signaling a dry run is therefore a desirable feature of the protocol.

Note that management logging, monitoring, alerts and tools for intervention are required. However, these can only be features of individual ASAs, not of the protocol itself. Another document [I-D.ietf-anima-stable-connectivity] discusses how such agents may be linked into conventional OAM systems via an Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane].

SN8. The protocol will be able to deal with a wide variety of technical objectives, covering any type of network parameter. Therefore the protocol will need a flexible and easily extensible format for describing objectives. At a later stage it may be desirable to adopt an explicit information model. One consideration is whether to adopt an existing information model or to design a new one.

E.3. Specific Technical Requirements

T1. It should be convenient for ASA designers to define new technical objectives and for programmers to express them, without excessive impact on run-time efficiency and footprint. In particular, it should be convenient for ASAs to be implemented independently of each other as user space programs rather than as kernel code, where such a programming model is possible. The classes of device in which the protocol might run is discussed in [I-D.ietf-anima-reference-model].

T2. The protocol should be easily extensible in case the initially defined discovery, synchronization and negotiation mechanisms prove to be insufficient.

T3. To be a generic platform, the protocol payload format should be independent of the transport protocol or IP version. In particular, it should be able to run over IPv6 or IPv4. However, some functions, such as multicasting on a link, might need to be IP version dependent. By default, IPv6 should be preferred.

T4. The protocol must be able to access off-link counterparts via routable addresses, i.e., must not be restricted to link-local operation.

T5. It must also be possible for an external discovery mechanism to be used, if appropriate for a given technical objective. In other words, GRASP discovery must not be a prerequisite for GRASP negotiation or synchronization.

T6. The protocol must be capable of distinguishing multiple simultaneous operations with one or more peers, especially when wait states occur.

T7. Intent: Although the distribution of Intent is out of scope for this document, the protocol must not by design exclude its use for Intent distribution.

T8. Management monitoring, alerts and intervention: Devices should be able to report to a monitoring system. Some events must be able to generate operator alerts and some provision for emergency intervention must be possible (e.g. to freeze synchronization or negotiation in a mis-behaving device). These features might not use the signaling protocol itself, but its design should not exclude such use.

T9. Because this protocol may directly cause changes to device configurations and have significant impacts on a running network, all protocol exchanges need to be fully secured against forged messages and man-in-the middle attacks, and secured as much as reasonably possible against denial of service attacks. There must also be an encryption mechanism to resist unwanted monitoring. However, it is not required that the protocol itself provides these security features; it may depend on an existing secure environment.

Appendix F. Capability Analysis of Current Protocols

This appendix discusses various existing protocols with properties related to the requirements described in Appendix E. The purpose is to evaluate whether any existing protocol, or a simple combination of existing protocols, can meet those requirements.

Numerous protocols include some form of discovery, but these all appear to be very specific in their applicability. Service Location Protocol (SLP) [RFC2608] provides service discovery for managed networks, but requires configuration of its own servers. DNS-SD [RFC6763] combined with mDNS [RFC6762] provides service discovery for small networks with a single link layer. [RFC7558] aims to extend this to larger autonomous networks but this is not yet standardized. However, both SLP and DNS-SD appear to target primarily application layer services, not the layer 2 and 3 objectives relevant to basic network configuration. Both SLP and DNS-SD are text-based protocols.

Simple Network Management Protocol (SNMP) [RFC3416] uses a command/response model not well suited for peer negotiation. Network Configuration Protocol (NETCONF) [RFC6241] uses an RPC model that does allow positive or negative responses from the target system, but this is still not adequate for negotiation.

There are various existing protocols that have elementary negotiation abilities, such as Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315], Neighbor Discovery (ND) [RFC4861], Port Control Protocol (PCP) [RFC6887], Remote Authentication Dial In User Service (RADIUS) [RFC2865], Diameter [RFC6733], etc. Most of them are configuration or management protocols. However, they either provide only a simple request/response model in a master/slave context or very limited negotiation abilities.

There are some signaling protocols with an element of negotiation. For example Resource ReSerVation Protocol (RSVP) [RFC2205] was designed for negotiating quality of service parameters along the path of a unicast or multicast flow. RSVP is a very specialised protocol aimed at end-to-end flows. A more generic design is General Internet Signalling Transport (GIST) [RFC5971], but it is complex, tries to solve many problems, and is also aimed at per-flow signaling across many hops rather than at device-to-device signaling. However, we cannot completely exclude extended RSVP or GIST as a synchronization and negotiation protocol. They do not appear to be directly useable for peer discovery.

RESTCONF [RFC8040] is a protocol intended to convey NETCONF information expressed in the YANG language via HTTP, including the ability to transit HTML intermediaries. While this is a powerful approach in the context of centralised configuration of a complex network, it is not well adapted to efficient interactive negotiation between peer devices, especially simple ones that might not include YANG processing already.

The Distributed Node Consensus Protocol (DNCP) [RFC7787] is defined as a generic form of state synchronization protocol, with a proposed usage profile being the Home Networking Control Protocol (HNCP) [RFC7788] for configuring Homenet routers. A specific application of DNCP for autonomic networking was proposed in [I-D.stenberg-anima-adncp].

DNCP "is designed to provide a way for each participating node to publish a set of TLV (Type-Length-Value) tuples, and to provide a shared and common view about the data published... DNCP is most suitable for data that changes only infrequently... If constant rapid state changes are needed, the preferable choice is to use an additional point-to-point channel..."

Specific features of DNCP include:

- o Every participating node has a unique node identifier.
- o DNCP messages are encoded as a sequence of TLV objects, sent over unicast UDP or TCP, with or without (D)TLS security.
- o Multicast is used only for discovery of DNCP neighbors when lower security is acceptable.
- o Synchronization of state is maintained by a flooding process using the Trickle algorithm. There is no bilateral synchronization or negotiation capability.
- o The HNCP profile of DNCP is designed to operate between directly connected neighbors on a shared link using UDP and link-local IPv6 addresses.

DNCP does not meet the needs of a general negotiation protocol, because it is designed specifically for flooding synchronization. Also, in its HNCP profile it is limited to link-local messages and to IPv6. However, at the minimum it is a very interesting test case for this style of interaction between devices without needing a central authority, and it is a proven method of network-wide state synchronization by flooding.

The Server Cache Synchronization Protocol (SCSP) [RFC2334] also describes a method for cache synchronization and cache replication among a group of nodes.

A proposal was made some years ago for an IP based Generic Control Protocol (IGCP) [I-D.chaparadza-intarea-igcp]. This was aimed at information exchange and negotiation but not directly at peer discovery. However, it has many points in common with the present work.

None of the above solutions appears to completely meet the needs of generic discovery, state synchronization and negotiation in a single solution. Many of the protocols assume that they are working in a traditional top-down or north-south scenario, rather than a fluid peer-to-peer scenario. Most of them are specialized in one way or another. As a result, we have not identified a combination of existing protocols that meets the requirements in Appendix E. Also, we have not identified a path by which one of the existing protocols could be extended to meet the requirements.

Authors' Addresses

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Email: cabo@tzi.org

Brian Carpenter (editor)
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

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

Email: leo.liubing@huawei.com

ANIMA WG S. Jiang, Ed.
Internet-Draft Z. Du
Intended status: Informational Huawei Technologies Co., Ltd
Expires: April 20, 2016 B. Carpenter
Univ. of Auckland
Q. Sun
China Telecom
October 18, 2015

Autonomic Prefix Management in Large-scale Networks
draft-jiang-anima-prefix-management-02

Abstract

This document describes an autonomic solution for prefix management in large-scale networks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document proposes an autonomic solution for prefix management in large-scale networks. The background to Autonomic Network (AN) is described in [RFC7575] and [RFC7576]. A generic autonomic signaling protocol (GRASP) is proposed by [I-D.ietf-anima-grasp], which would be used by the proposed autonomic prefix management solution.

This document is dedicated to how to make IPv6 prefix management in pure IPv6 large-scale networks as autonomic as possible. This document for now is only considering service provider (ISP) networks. Although there are similarities with large enterprise networks, the requirements are a little different for the two use cases.

Note in draft: This version is preliminary. In particular, many design details may be subject to change until the anima specifications become agreed.

2. Problem Statement

The autonomic networking use case considered here is autonomic IP address management in large-scale networks.

Although DHCPv6 Prefix Delegation [RFC3633] has supported automated delegation of IPv6 prefixes, prefix management is still largely depending on human planning. In other words, there is no basic information or policy to support autonomic decisions on the prefix length that each router should request or be delegated, according to its role in the network. Roles could be locally defined or could be generic (edge router, interior router, etc.). Furthermore, the current IPv6 prefix management by humans is rigid and static after initial planning.

The problem to be solved by AN is how to dynamically and autonomically manage IPv6 address space in large-scale networks, so that IPv6 addresses can be used efficiently. The AN approach discussed in this document is based on the assumption that there is a generic discovery and negotiation protocol that enables direct negotiation between intelligent IP routers. [I-D.ietf-anima-grasp] is one of the attempts at such a protocol.

2.1. Intended User and Administrator Experience

The intended experience is, for the administrator(s) of a large-scale network, that the management of IPv6 address space can be run with minimum efforts, for both the network and network device initiation stage and during running time. In the ideal scenario, the administrator(s) only have to configure a single IPv6 prefix for the whole network and the initial prefix length for each device role.

The actual address usage needs to be logged for potential offline management operations including audit and security incident tracing.

2.2. Analysis of Parameters and Information Involved

For specific purposes of address management, a few parameters are involved on each device (some of them can be pre-configured before they are connected). They include:

- Identity of this device. It can be verified by the certification authority (CA) that is maintained by the network administrator(s).

- Identity of a trust anchor which is certification authority (CA) that is maintained by the network administrator(s).

- Role of this device.

- o An IPv6 prefix length for this device.

- o An IPv6 prefix that is assigned to this device and its downstream devices.

A few parameters are involved in the network as a whole. They are:

- o Identity of a trust anchor which is a certification authority (CA) that is maintained by the network administrator(s).

- o Total IPv6 address space. It is one (or several) IPv6 prefix(es).

- o The initial prefix length for each device role.

2.2.1. Parameters each device can decide for itself

This section identifies those of the above parameters that do not need external information in order for the devices concerned to set them to a reasonable value after bootstrap or after a network disruption. There are few of these:

- o Role of this device.

- o Default IPv6 prefix length for this device.

- o Identity of this device.

The device may be shipped from the manufacture with pre-configured role and default prefix length.

2.2.2. Information needed from policy intent

This section identifies those parameters that need external information about policy intent in order for the devices concerned to set them to a non-default value.

- o Non-default value for the IPv6 prefix length for this device. This needs to be decided based on the role of this device.

- o The initial prefix length for each device role.

- o Identity of a trust anchor.

- o Whether to allow the device request more address space.

- o The policy when to request more address space, for example, the address usage reaches a certain limit or percentage.

2.2.3. Comparison with current solutions

This section briefly compares the above use case with current solutions. Currently, the address management is still largely depending on human planning. It is rigid and static after initial planning. The address requests will fail if the configured address space is used up.

Some functions, for autonomic and dynamic address management, may be achievable by extending the existing protocols, for example, extending DHCPv6-PD to request IPv6 address according to the device role. However, defining uniform device roles may not be a practical task. Some functions are not suitable to be achieved by any existing protocols.

However, using a generic autonomic discovery and negotiation protocol instead of specific solutions has the advantage that additional parameters can be included in the autonomic solution without creating new mechanisms. This is the principal argument for a generic approach.

2.3. Interaction with other devices

2.3.1. Information needed from other devices

This section identifies those of the above parameters that need external information from neighbor devices (including the upstream devices). In many cases, two-way dialogue with neighbor devices is needed to set or optimize them.

- o Identity of a trust anchor.
- o The device will need to discover a device, from which it can acquire IPv6 address space.
- o The initial prefix length for each device role, particularly for its own downstream devices.
- o The default value of the IPv6 prefix length may be overridden by a non-default value.
- o The device will need to request and acquire IPv6 prefix that is assigned to this device and its downstream devices.
- o The device may respond to prefix delegation request from its downstream devices.

o The device may require to be assigned more IPv6 address space, if it used up its assigned IPv6 address space.

2.3.2. Monitoring, diagnostics and reporting

This section discusses what role devices should play in monitoring, fault diagnosis, and reporting.

o The actual address assignments need to be logged for the potential offline management operations.

o In general, the usage situation of address space should be reported to the network administrators, in an abstract way, for example, statistics or visualized report.

o A forecast of address exhaustion should be reported.

3. Autonomic Prefix Management Solution

This section introduces an autonomic prefix management solution. It extends the generic discovery and negotiation protocol defined by [I-D.ietf-anima-grasp]. The relevant options are defined in Section 4.

3.1. Behaviors to discover prefix providing device

A device should decide the length of request prefix by the intent-based mechanism, described in Section 5. If it used up its current address resource, it could request more, which is not necessary to be on the same scale as its initial resource.

A prefix requesting device that needs new or more address space should firstly discover peer devices that may be able to provide extra address space. The device should send out a GRASP Discovery message that contains a Prefix Objective option Section 4.1, in which the device also indicates whether it supports the DHCPv6 Prefix Delegation (PD) [RFC3633] function and the length of requested prefix.

3.2. Behaviors on prefix providing device

A peer device receiving a Discovery message with a Prefix Objective option, if it is able to provide such a prefix, should respond with a GRASP Response message. The Response message also carries a Prefix Objective option, which also indicate whether the peer device supports the PD function and the available prefix length matching the request. If the peer device does not have enough resource, it may silently drop the Discovery message or return a GRASP Response message, which contains a longer prefix length (smaller address space) that it can provide. A divert option may also be added into the GRASP Response message. This divert option indicates another device that may provide the prefix. The diverted device is typically an upstream gateway router, but it could in theory be any device that might have unused prefix space.

A gateway router in a hierarchical network topology is normally responsible to provide prefixes for routers within its subnet. In the case that it does not have enough resource for the downstream requesting router, it should return a GRASP Response message, which contains a longer prefix length (smaller address space) that this gateway router may provide. In this case too, a divert option may be added into the GRASP Response message. The diverted device is typically another upstream gateway router.

A resource shortage may cause the gateway router to request more resource from its upstream device. This would be another independent GND discovery and negotiation process. During the processing time, the gateway router should send a Confirm-waiting Message to the initial requesting router. When the new resource becomes available, the gateway router responds with a GRASP Response message with the prefix length matching the request.

The algorithm to choose which prefixes to assign on the prefix providing devices is an implementation choice out of document scope.

3.3. Prefix Requests Behaviors

Upon receiving the GRASP Response message that indicates the requesting prefix length is accepted, the requesting device may request the prefix using DHCPv6 PD, if both itself and the response device support PD.

Upon receiving the GRASP Response message that indicates the requesting prefix length is not possible, but a longer prefix length is available, the requesting device may request the longer prefix using DHCPv6 PD, if both itself and the response device support PD.

If the GRASP Response message carries a divert option, the requesting device may sent an unicast GRASP Discovery message to the diverted device to find out whether that device can provide the requested length prefix.

[Author’s note: undecided whether we should support prefix delegation using the GRASP protocol. This would have some partial overlap with DHCPv6 PD. But it seems more consistent as a solution.]

3.4. Prefix log

Within the autonomic prefix management, all the prefix assignment is done by devices without human intervention. It is even more important to record all the prefix assignment history. However, the logging and reporting process is out of document scope.

4. Autonomic Prefix Management Options

This section defines the GRASP options that are used to support autonomic prefix management.

4.1. Prefix Objective option

The Prefix Objective option carries the PD support flag and the prefix length. The format of the Prefix Objective option is described as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Prefix_Obj_Option        |         option-len            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|PD_Support_Flag| Prefix_Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
option-code    Prefix_Obj_Option (TBA1).

option-len     2, length of option content in octets.

PD_Support_Flag Indicates whether the message sender supports
               DHCPv6 Prefix Delegation function, 1 for support,
               0 for no support, as client or server accordingly.
               This flag must not be set to any other values.

Prefix_Length  Indicate the prefix length that the message sender
               requests or is willing to provide.
```

5. Prefix Management Intent

With in a single administrative domain, the network operator could manage all their devices with role set. If so, there is possibility to configure/manage the prefix length for every device in a simple way.

The network operator could only manage the default prefix length for each type of role. A prefix management intent, which contains all mapping information of device roles and their default prefix lengths, should be flooded in the network, through the Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. The intent flooding mechanism is out of document scope.

Upon receiving the prefix management intent, every device can decide its default prefix length by matching its own role.

5.1. Example of Prefix Management Intent

The prefix management intent in this document is used to carry mapping information of device roles and their default prefix lengths in an autonomic domain. For example, an IPRAN operator wants to configure the prefix length of RNC Site Gateway (RSG) as 34, the prefix length of Aggregation Site Gateway (ASG) as 44, and the prefix length of Cell Site Gateway (CSG) as 56. She/he may input the following intent into the autonomic network:


```
{"autonomic_intent":
[
   {"model_version": "1.0"},
   {"intent_type": "Network management"},
   {"autonomic_domain": "Customer_X_intranet"},
   {"intent_name": "Prefix management"},
   {"intent_version": 73},
   {"Timestamp": "20150606 00:00:00"},
   {"Lifetime": "Permanent"},
   {"signature": "XXXXXXXXXXXXXXXXXXX"},
   {"content":
   [
      {"role": [{"role_name": "RSG"},
         {"role_characteristic":
            [{"prefix_length": "34"}]}
         ]},
      {"role": [{"role_name": "ASG"},
         {"role_characteristic":
            [{"prefix_length": "44"}]}
         ]},
      {"role": [{"role_name": "CSG"},
         {"role_characteristic":
            [{"prefix_length": "56"}]}
         ]}
   ]
   }
]
}
```

6. Security Considerations

Relevant security issues are discussed in [I-D.ietf-anima-grasp]. The security mechanism in this document is established on a Public Key Infrastructure (PKI) system [RFC3647] that is maintained by the network administrator(s).

It is RECOMMENDED that DHCPv6 PD, if used, should be operated using DHCPv6 authentication or Secure DHCPv6.

7. IANA Considerations

This document defines one new GRASP option. The IANA is requested to assign a value for this option from the GRASP Option Codes table of the GRASP Parameters registry as defined by [I-D.ietf-anima-grasp] (if approved).

- o The Prefix Objective option (TBA1), described in Section 4.1.

8. Acknowledgements

Valuable comments were received from Michael Behringer and Chongfeng Xie.

This document was produced using the xml2rfc tool [RFC2629].

9. Change log [RFC Editor: Please remove]

draft-jiang-anima-prefix-management-00: original version, 2014-10-25.

draft-jiang-anima-prefix-management-01: add intent example and coauthor Zongpeng Du, 2015-05-04.

draft-jiang-anima-prefix-management-02: update references and the format of the prefix management intent, 2015-10-14.

10. References

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-01 (work in progress), October 2015.

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-01 (work in progress), October 2015.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <http://www.rfc-editor.org/info/rfc3633>.

[RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, DOI 10.17487/RFC3647, November 2003, <http://www.rfc-editor.org/info/rfc3647>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7576>.

Authors' Addresses

Sheng Jiang (editor)
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

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

Email: duzongpeng@huawei.com

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

Email: brian.e.carpenter@gmail.com

Qiong Sun
China Telecom
No.118, Xizhimennei Street
Beijing 100035
P. R. China

Email: sunqiong@ctbri.com.cn

ANIMA WG B. Liu
INTERNET-DRAFT Huawei Technologies
Intended Status: Standard Track X. Xiao
Expires: May 7, 2020 MRC, Huawei Technologies
S. Jiang
Huawei Technologies
A. Hecker
Z. Despotovic
MRC, Huawei Technologies
November 4, 2019

Information Distribution in Autonomic Networking
draft-liu-anima-grasp-distribution-12

Abstract

This document discusses the requirement to autonomic nodes supporting information distribution based over GRASP in autonomic networks. In general, information distribution can be categorized into two different modes: 1) one autonomic node instantly sends information to other nodes in the domain; 2) one autonomic node publishes some information and asynchronously some other interested nodes request the published information. In the former case, information data will be generated and consumed instantly. In the latter case, information data live longer in the network.

These capabilities are basic and fundamental to an autonomous network system (i.e. ANI [I-D.ietf-anima-reference-model]). This document clarifies possible use cases of information distribution in ANI and requirements to ANI so that rich information distribution can be natively supported. Extensions to autonomic nodes are proposed and detailed embodiments based on GRASP are discussed.

Status of this Memo

Copyright and License Notice

Table of Contents

1 Introduction

In an autonomic network, autonomic functions (AFs) running on autonomic nodes would exchange information constantly, both for controlling/management signaling and data exchange. This document discusses the information distribution capability of such exchanges between AFs.

According to the number of participants, information distribution can happen with the following scenarios:

1) Point-to-point (P2P) Communication: information are exchanged between two communicating parties from one node to another node.

2) One-to-Many Communication: information exchanges involve an information source and multiple receivers.

The approaches of distributing information could be categorized into two basic modes:

1) An instant communication: a sender connects and sends the information data (e.g. control/management signaling, synchronization data and so on) to the receiver(s) immediately.

2) An asynchronous communication: a sender saves the information in the network, may or may not publish the information to the other who is interested in the published information, to which a node asks to retrieve.

The ANI should have provided a generic way to support these information distribution scenarios among AFs, rather than AFs managing to use other transport or routing protocols (HTTP, BGP/IGP as bearing protocols etc.). In fact, GRASP already provides part of the capabilities.

In this document, we first analyze requirements of information distribution in autonomic networks (Section 3), and then introduce its relationship to the other modules in ANI (Section 4). After that, the node behaviors and extensions to the existing GRASP are introduced in Section 5 and Section 6, respectively.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Requirements of Advanced Information Distribution

If the information exchanged is just short and simple, this can be done instantly. In practice, however, this is not always the case. In the following cases, a mixture of instant and asynchronous communication models is more appropriate.

1) Long Communication Intervals. The time interval of the communication is not necessarily always short and instant. Advanced AFs may rather involve heavy jobs/tasks (e.g. database lookup, authentication etc.) when gearing the network, so the instant mode may introduce unnecessary pending time and become less efficient. If simply using an instant mode, the AF has to wait until the tasks finish and return. A better way is that an AF instantly sends the request but switches to an synchronous mode, once the jobs are finished, AFs will get notified.

2) Common Interest Distribution. As mentioned, some information are common interests among AFs. For example, the network intent is distributed to network nodes enrolled, which is a typical one-to-many scenario. We can also finish the intent distribution by an instant flooding (e.g. via GRASP) to every network nodes across the network domain. Because of network dynamic, however, not every node can be just ready at the moment when the network intent is flooded. Actually, nodes may join in the network sequentially. In this situation, an asynchronous communication model could be a better choice where every (newly joining) node can subscribe the intent information and will get notified if it is ready (or updated).

3) Distributed Coordination. With computing and storage resources on autonomic nodes, alive AFs not only consumes but also generates data information. For example, AFs coordinating with each other as distributed schedulers, responding to service requests and distributing tasks. It is critical for those AFs to make correct decisions based on local information, which might be asymmetric as well. AFs may also need synthetic/aggregated data information (e.g. statistic info, like average values of several AFs, etc.) to make decisions. In these situations, AFs will need an efficient way to form a global view of the network (e.g. about resource consumption, bandwidth and statistics). Obviously, purely relying on instant communication model is inefficient, while a scalable, common, yet distributed data layer, on which AFs can store and share information in an asynchronous way, should be a better

choice.

For ANI, in order to support various communication scenarios, an information distribution module is required, and both instant and asynchronous communication models should be supported.

4. Information Distribution Module in ANI

This section describes how the information distribution module fits into the ANI including what extensions of GRASP are required [I-D.ietf-anima-grasp].

```
                         +--------------------+
                         |        ASAs        |
                         +--------------------+
                                  ^
                                  |
   v
    +------------------------Info-Dist. APIs------------------------+
    | +---------------------+   +--------------+   +----------------+ |
    | | Selective Flooding  |-|-| Event Queue  |-|-| Info. Storage  | |
    | +---------------------+   +--------------+   +----------------+ |
    +---------------------------------------------------------------+
                               ^
                               |
                               v
                +-------------GRASP APIs-----------------+
                | +---------------+   +----------------+ |
                | |  GRASP Base   |-|-|   Extension  | | |
                | +---------------+   +----------------+ |
                +----------------------------------------+
```

Figure 1. Information Distribution Module and GRASP Extension.

As the Fig 1 shows, the information distribution module includes three sub-modules, all of which provides APIs for ASAs. Specific behaviors of these modules is described in Section 5. In order to support the modules, the GRASP is also extended, which is described in Section 6.

5. Node Behaviors

In this section, we discuss how each autonomic node should behave in order to support the two modes of information distribution introduced before. ANI is a distributed system, so the information distribution module must be implemented in a distributed way as well, where every node participates to contribute.

5.1 Instant Information Distribution

In this case, sender(s) and receiver(s) are specified. Information will be directly sent from the sender(s) to the receiver(s). This requires that every node is equipped by some signaling/transport protocols so that they can coordinate with each other and correctly deliver the information.

5.1.1 Instant P2P and Flooding Communications

Current GRASP provides the capability to support instant P2P and flooding. It is natural to use the GRASP Synchronization message directly for P2P distribution. Furthermore, it is also natural to use the GRASP Flood Synchronization message for broadcast.

However, as mentioned in Section 3, in some scenarios one node needs to communicate some information to another, rather than synchronization. As a result, an un-solicited synchronization mechanism is needed in GRASP. An extension of GRASP message (i.e. M_UNSOLIDSYN) is defined in Section 6.1.

5.1.2 Instant Selective Flooding Communication

When doing selective flooding, the distributed information needs to contain criteria for nodes to judge which interfaces should be sent the distributed information and which are not. Specifically, the criteria contain:

- o Matching Condition: a set of matching rules.

- o Matching Object: the object that the match condition would be applied to. For example, the matching object could be node itself or its neighbors.

- o Action: what behavior the node needs to do when the matching object matches or failed the matching condition. For example, the action could be forwarding or discarding the distributed message.

The criteria information must be included in the message distributed from the sender. The receiving node decides its reaction according to the criteria carried in the message. Given the criteria appended with the message, the node behaviors can be:

- o When the Matching Object is "Neighbors", then the node matches the relevant information of its neighbors to the Matching Condition. If the node finds one neighbor matches the Matching Condition, then it forwards the message to the neighbor. If not, the node discards the message.

o When the Matching Object is the node itself, then the node matches against to the Matching Condition. If the node finds itself matches the Matching Condition, then it forwards the distributed message to its neighbors; if not, the node discards the message.

GRASP is extended with a new option field (i.e. selective-flood-option) to support selective flooding, shown in Section 6.2. This option will be included in the original flooding message of GRASP. An example of selective flooding is briefly described in the Appendix A.

5.2 Asynchronous Information Distribution

Asynchronous information distribution happens in a different way where sender(s) and receiver(s) are not immediately specified while they may appear in an asynchronous way. First of all, this requires that the information can be stored in the network; secondly, it requires an information publication and subscription (Pub/Sub) mechanism (corresponding protocol specification of Pub/Sub is defined in Section 6).

As we sketched in the previous section that in general each node requires two modules: 1) Information Storage (IS) module and 2) Event Queue (EQ) module in the information distribution module. We introduce details of the two modules in the following sections.

5.2.1 Information Storage

IS module handles how to save and retrieve information for ASAs across the network. The IS module uses a syntax to index information, generating the hash index value (e.g. a 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. Such a request will be checked by the IS module who will be responsible for the request whether such a request is feasible according to criteria such as permitted information size.

2) Destination Node Mapping. The information block will be handled by the IS module in order to calculate/map to a destination node in the network. Since ANIMA network is a peer-to-peer network, a typical way is to use dynamic hash table (DHT) to map information to a unique index identifier. For example, if the size of the

information is reasonable, the information block itself can be hashed, otherwise, some meta-data of the information block can be used to generate the mapping.

3) Destination Node Negotiation Request. Negotiation request of storing the information will be sent from the IS module to the IS module on the destination node. The negotiation request contains parameters about the information block from the source IS module. According to the parameters as well as the local available resource, the destination node will feedback the source IS module.

4) Destination Node Negotiation Response. Negotiation response from the destination node is sent back to the source IS module. If the source IS module gets confirmation that the information can be stored, source IS module will prepare to transfer the information block; otherwise, a new destination node must be discovered (i.e. going to step 7).

5) Information Block Transfer. Before sending the information block to the destination node that accepts the request, the IS module of the source node will check if the information block can be afforded by one GRASP message. If so, the information block will be directly sent by calling a GRASP API. Otherwise, bulk data transmission with GRASP will be triggered, where multi-time GRASP message sending will be used so that one information block will be transferred by smaller pieces [I-D.ietf-anima-reference-model].

6) Information Writing. Once the information block (or a smaller block) is received, the IS module of the destination node will store the data block in the local storage, which is accessible.

7) (Optional) New Destination Node Discovery. If the previously selected destination node is not available to store the information block, the source IS module will have to identify a new destination node to start a new negotiation. In this case, the discovery can be done by using discovery GRASP API to identify a new candidate, or more complex mechanisms can be introduced.

Similarly, Getting information from an ANIMA network will be realized in the following steps.

1) ASA-to-IS Request. An ASA accesses the IS module via the APIs exposed by the information distribution module. The key/index of the interested information will be sent to the IS module. An assumption here is that the key/index should be ready to an ASA before an ASA can ask for the information. This relates to the publishing/subscribing of the information, which are handled by other modules (e.g. Event Queue with Pub/Sub supported by GRASP).

2) Destination Node Mapping. IS module maps the key/index of the requested information to a destination node, and prepares to start to request the information. The mapping here follows the same mechanism when the information is stored.

3) Retrieval Negotiation Request. The source IS module sends a request to the destination node identified in the previous step and asks if such an information object is available.

4) Retrieval Negotiation Response. The destination node checks the key/index of the requested information, and replies to the source IS module. If the information is found and the information block can be afforded within one GRASP message, the information will be sent together with the response to the source IS module.

5) (Optional) New Destination Request. If the information is not found after the source IS module gets the response from the original destination node, the source IS module will have to discover where the location storing the requested information is.

IS module can reuse distributed databases and key value stores like NoSQL, Cassandra, DHT technologies. storage and retrieval of information are all event-driven responsible by the EQ module.

5.2.2 Event Queue

The main job of Event Queue (EQ) module is to help ASAs to subscribe interested information and publish information to the network in asynchronous scenarios. In 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/subscribe.

3) Event Matching: an information consumer has to be identified from the queue in order to deliver the information from the provider. Event matching keeps looking for the subscriptions in the queue to

see if there is an exact published event there. Whenever a match is found, it will notify the upper layer to inform the corresponding ASAs who are the information provider and subscriber(s) respectively.

The procedure of how EQ module on every network node works is introduced as follows.

1) Event ID Generation: If information of an ASA is ready, an event ID is generated according to the content of the information. This is also related to how the information is stored/saved by the IS module introduced before. Meanwhile, the type of the event is also specified where it can be of control purpose or user plane data.

2) Priority Specification: According to the type of the event, the ASA may specify its priority to say how this event is wanted to be processed. By considering both aspects, the priority of the event will be determined and ready for enqueuing.

3) Event Enqueue: Given the event ID, event group and its priority, a queue is identified locally if all criteria can be satisfied. If there is such a queue, the event will be simply added into the queue, otherwise a new queue will be created to accommodate such an event.

4) Event Propagation: The published event will be propagated to the other network nodes in the ANIMA domain. A propagation algorithm can be employed to here in order to optimize the propagation efficiency of the updated event queue states.

5) Event Match and Notification: While propagating updated event states, EQ module in parallel keeps matching published events and its interested consumers. Once a match is found, the provider and subscriber(s) will be notified for final information retrieval.

Here we define the category of event priority. In general, we define two event types:

1) Network Control Event: This type of events are defined by the ANI for operational purposes on network control. We suggest a pre-defined priority levels for required system messages. 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 in 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.

5.2.3 Integrating 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 e published to the network. For this, GRASP APIs discover(), synchronize() and flood() are combined to compose such a procedure. In specific, discover() call will specific 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.

5.3 Summary

In summary, the general requirements for the information distribution module on each autonomic node are two sub-modules handling instant communications and asynchronous communications, respectively. For instant communications, node requirements are simple, in which signaling protocols have to be supported. With minimum efforts, reusing the existing GRASP is possible. For asynchronous communications, information distribution module requires event queue and information storage mechanism to be supported.

6. Protocol Specification (GRASP extension)

There are multiple ways to integrate the information distribution module. The principle we follow is to minimize modifications made to the current ANI. We consider to use GRASP as a base to build up the information distribution module. The main reason is that the current version of GRASP is already an information distribution module for the cases of P2P and flooding. In the following discussions, we introduce how to complete the missing part.

6.1 Un-solicited Synchronization Message (A new GRASP Message)

In fragmentary CDDL, a Un-solicited Synchronization message follows the pattern:

```
unsolicited_synch-message = [M_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).

6.2 Selective Flooding Option

In fragmentary CDDL, the selective flood 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 selective flood option encapsulates a match-condition option which represents the conditions regarding to continue or discontinue flood the current message. For the match-condition option, the Obj1 and Obj2 are to objects that need to be compared. For example, the Obj1 could be the role of the device and Obj2 could be "RSG". The match rules between the two objects could be greater, less than, within, or contain. The match-object represents of which Obj1 belongs to, it could be the device itself or the neighbor(s) intended to be flooded. The action means, when the match rule applies, the current device just continues flood or discontinues.

6.3 Subscription Objective Option

In fragmentary CDDL, a Subscription Objective Option follows the pattern:

```
      subscription-objection-option = [SUBSCRIPTION, 2, 2, subobj]
      objective-name = SUBSCRIPTION
      objective-flags = 2
      loop-count = 2
      subobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a subscription to a specific object.

6.4 Un_Subscription Objective Option

In fragmentary CDDL, a Un_Subscribe Objective Option follows the pattern:

```
      Unsubscribe-objection-option = [UNSUBSCRIB, 2, 2, unsubobj]
      objective-name = SUBSCRIPTION
      objective-flags = 2
      loop-count = 2
      unsubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a un-subscription to a specific object.

6.5 Publishing Objective Option

In fragmentary CDDL, a Publish Objective Option follows the pattern:

```
publish-objection-option = [PUBLISH, 2, 2, pubobj] objective-name
= PUBLISH
objective-flags = 2
loop-count = 2
pubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a publish of a specific object data.

Note that extended GRASP messages with new arguments inside here will be triggered by interactions/actions from information distribution module introduced in Section 5.

7. 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.

8. IANA Considerations

TBD.

9. References

9.1 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.

9.2 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-02]
Carpenter, B., Jiang, S., Liu, B., "Transferring Bulk Data over the GeneRic Autonomic Signaling Protocol (GRASP)", draft-carpenter-anima-grasp-bulk-02 (work in progress), June 30, 2018

[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/>

Appendix A.

GRASP includes flooding criteria together with the delivered information so that every node will process and act according to the criteria specified in the message. An example of extending GRASP with selective criteria can be:

- o Matching condition: "Device role=IPRAN_RSG"
- o Matching objective: "Neighbors"
- o Action: "Forward"

This example means: only distributing the information to the neighbors who are IPRAN_RSG.

Authors' Addresses

Bing Liu
Huawei Technologies
Q27, Huawei Campus
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 of as an infrastructure layer module, which provides to upper layer utilizations. In this section, we review some use cases from the real-world where an information distribution module with powerful functions do plays a critical role there.

A.1 Service-Based Architecture (SBA) in 3GPP 5G

In addition to Internet, the telecommunication network (i.e. carrier mobile wireless networks) is another world-wide networking system.

The architecture of the upcoming 5G mobile networks from 3GPP has already been defined to follow a service-based architecture (SBA) where any network function (NF) can be dynamically associated with any other NF(s) when needed to compose a network service. Note that one NF can simultaneously associate with multiple other NFs, instead of being physically wired as in the previous generations of mobile networks. NFs communicate with each other over service-based interface (SBI), which is also standardized by 3GPP [3GPP.23.501].

In order to realize an SBA network system, detailed requirements are further defined to specify how NFs should interact with each other with information exchange over the SBI. We now list three requirements that are related to information distribution here.

1) NF Pub/Sub: Any NF should be able to expose its service status to the network and any NF should be able to subscribe the service status of an NF and get notified if the status is available. An concrete example is that a session management function (SMF) can subscribe the REGISTER notification from an access management function (AMF) if there is a new user entity trying to access the mobile network [3GPP.23.502].

2) Network Exposure Function (NEF): A particular network function that is required to manage the event exposure and distributions. In specific, SBA requires such a functionality to register network events from the other NFs (e.g. AMF, SMF and so on), classify the events and properly handle event distributions accordingly in terms of different criteria (e.g. priorities) [3GPP.23.502].

3) Network Repository Function (NRF): A particular network function where all service status information is stored for the whole network. An SBA network system requires all NFs to be stateless so as to improve the resilience as well as agility of providing network services. Therefore, the information of the available NFs and the service status generated by those NFs will be globally stored in NRF as a repository of the system. This clearly implies storage capability that keeps the information in the network and provides those information when needed. A concrete example is that whenever a new NF comes up, it first of all registers itself at NRF with its profile. When a network service requires a certain NF, it first inquires NRF to retrieve the availability information and decides whether or not there is an available NF or a new NF must be instantiated [3GPP.23.502].

(Note: 3GPP CT might finally adopt HTTP2.0/JSON to be the protocol communicating between NFs, but autonomic networks can also load HTTP2.0 with in ACP.)

A.2 Vehicle-to-Everything

Carrier networks On-boarding services of vertical industries are also one of some blooming topics that are heavily discussed. Connected car is clearly one of the important scenarios interested in automotive manufacturers, carriers and vendors. 5G Automotive Alliance - an industry collaboration organization defines many promising use cases where services from car industry should be supported by the 5G mobile network. Here we list two examples as follows [5GAA.use.cases].

1) Software/Firmware Update: Car manufacturers expect that the software/firmware of their car products can be remotely updated/upgraded via 5G network in future, instead of onsite visiting their 4S stores/dealers offline as nowadays. This requires the network to provide a mechanism for vehicles to receive the latest software updates during a certain period of time. In order to run such a service for a car manufacturer, the network shall not be just like a network pipe anymore. Instead, information data have to be stored in the network, and delivered in a publishing/subscribing fashion. For example, the latest release of a software will be first distributed and stored at the access edges of the mobile network, after that, the updates can be pushed by the car manufacturer or pulled by the car owner as needed.

2) Real-time HD Maps: Autonomous driving clearly requires much finer details of road maps. Finer details not only include the details of just static road and streets, but also real-time information on the road as well as the driving area for both local urgent situations and intelligent driving scheduling. This asks for situational awareness at critical road segments in cases of changing road conditions. Clearly, a huge amount of traffic data that are real-time collected will have to be stored and shared across the network. This clearly requires the storage capability, data synchronization and event notifications in urgent cases from the network, which are still missing at the infrastructure layer.

A.3 Summary

Through the general analysis and the concrete examples from the real-world, we realize that the ways information are exchanged in the coming new scenarios are not just short and instant anymore. More advanced as well as diverse information distribution capabilities are required and should be generically supported from the infrastructure layer. Upper layer applications (e.g. ASAs in ANIMA) access and utilize such a unified mechanism for their own services.

ANIMA P. Peloso
Internet-Draft L. Ciavaglia
Intended status: Standards Track Nokia
Expires: September 22, 2016 March 21, 2016

A Day in the Life of an Autonomic Function
draft-peloso-anima-autonomic-function-01.txt

Abstract

While autonomic functions are often pre-installed and integrated with the network elements they manage, this is not a mandatory condition. Allowing autonomic functions to be dynamically installed and to control resources remotely enables more versatile deployment approaches and enlarges the application scope to virtually any legacy equipment. The analysis of autonomic functions deployment schemes through the installation, instantiation and operation phases allows constructing a unified life-cycle and identifying new required functionality. Thus, the introduction of autonomic technologies will be facilitated, the adoption much more rapid and broad. Operators will benefit from multi-vendor, inter-operable autonomic functions with homogeneous operations and superior quality, and will have more freedom in their deployment scenarios.

Requirements Language

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].

Copyright Notice

Table of Contents

1. Problem statement

While autonomic functions are often pre-installed and integrated with the network elements they manage, this is not a mandatory condition. Allowing autonomic functions to be dynamically installed and to control resources remotely enables more versatile deployment approaches and enlarges the application scope to virtually any legacy equipment. The analysis of autonomic functions deployment schemes through the installation, instantiation and operation phases allows constructing a unified life-cycle and identifying new required functionality.

An Autonomic Service Agent (ASA) controls resources of one or multiple Network Elements (NE), e.g. the interfaces of a router for a Load Balancing ASA. An ASA is a software, thus an ASA need first to be installed and to execute on a host machine in order to control resources.

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 provide the operator a great variety of ASA deployment schemes as they decorrelate the evolution of the infrastructure layer from the evolution of the autonomic function layer. Depending on the capabilities (and constraints) of the infrastructure and of the autonomic functions, the operator can devise the schemes that will better fit to its deployment objectives and practices.

Based on the above definitions, the ASA deployment process can be formulated as a multi-level/criteria matching problem.

The primary level, present in the three phases, consists in matching the objectives of the operator and the capabilities of the infrastructure. The secondary level criteria may vary from phase to phase. One goal of the document is thus to identify the specific and common functionality among these three phases.

This draft will explore the implications of these properties along each of the 3 phases namely Installation, Instantiation and Operation. This draft will then provide a synthesis of these implications in requirements for functionalities and life-cycle of ASAs. Beforehand, the following section will deal with the network operator’s point of view with regards of autonomic networks.

2. Motivations from an operator viewpoint

Only few operators would dare relying on a pure autonomic network, without setting objectives to it. From an operator to the other, the strategy of network management vary, as much for historical reasons (experience, best-practice, tools in-place, organization), as much for differences in the operators goals (business, trade agreements, politics, risk policy). Additionally, network operators do not necessarily perform a uniform network management across the different domains composing their network infrastructure. Hence their objectives in terms of availability, load, and dynamics vary depending on the nature of the domains and of the types of services running over each of those domains.

To manage the networks according to the above variations, ASAs need to capture the underlying objectives implied by the operators. The instantiation phase is the step in-between installation and operation, where the network operator determine the initial ASA behavior according to its objectives. This step allows the network operator to determine which ASAs should execute on which domains of its network, with appropriate settings. At this stage, thanks to the intent-policy setting objectives to groups of ASAs, the network management should become far simpler (and less error-prone) than setting low-level configurations for each individual network resources.

2.1. Illustration of increasingly constraining operator’s objectives

This paragraph describes the following example of operator intents with regards to deployments of autonomic functions. The autonomic function involved is a load balancing function, which uses monitoring results of links load to autonomously modify the links metrics in order to balance the load over the network. The example is divided into steps corresponding to an increasing implication of the operator in the definition of objectives/intents to the deployment of autonomic functions:

Step 1 The operator operates its network and benefits from the autonomic function on the nodes which have the installed ASAs.

Step 2 Then the operator, specifies to the autonomic function an objective which is to achieve the maximum number of links with a load below 6O%.

Step 3 The network is composed of five domains, a core transport network and four metropolitan networks, each interconnected through the core network, the operator sets a different objective to the autonomic function for each of the five domain.

Step 4 As inside metropolitan domains the traffic variations are steeper and happen in a periodic fashion contrary to the traffic in the core domain, the network operators installs an additional autonomic function inside each of these domains. This autonomic function is learning the traffic demands in order to predict traffic variations. The operators instructs the load balancing function to augment its monitored input with the traffic predictions issued by the newly installed autonomic function.

Step 5 As the algorithm of the load balancing autonomic function is relying on interactions between autonomic function agents, the operator expects the interactions to happen in-between ASAs of each domain, hence the load will be balanced inside each of the domain, while previously it would have been balanced over the whole network uniformly.

Step 6 Finally, the network operator has purchased a new piece of software corresponding to an autonomic function achieving load balancing with a more powerful algorithm. For trial sake, he decides to deploy this new load balancing function instead of the previous one on one of its four metropolitan domains.

This short example illustrates some specificities of deployment scenarios, the sub-section below sets itself at providing a more exhaustive view of the different deployment scenarios.

2.2. Deployment scenarios of autonomic functions

The following scenarios illustrate the different ways the autonomic functions could be deployed in an ANIMA context. Subsequently, requirements for the autonomic functions and requirements these autonomic functions impose on other components of the ANIMA ecosystem are listed.

These various deployment scenarios are better understood by referring to the High level view of an Autonomic Network, Figure 1 of

[I-D.behringer-anima-reference-model]. The figure is slightly extended for the purpose of the demonstration as follows:

```
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|            :      Autonomic Function 1       :            |
|
|  ASA 1.1   :      ASA 1.2   :   ASA 1.3      :   ASA 1.4  |
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
             :                :                :
             :  + - - - - - - - - - - - - - +  :
             :  |   Autonomic Function 2    |  :
             :  |                           |  :
             :  |   ASA 2.2   :   ASA 2.3   |  :
             :  + - - - - - - - - - - - - - +  :
             :                :                :
+ - - - - - - - - - - - - - + : + - - - - - - - - - - - - - +
|   Autonomic Function 3    | : |   Autonomic Function 4    |
|                           | : |                           |
|   ASA 3.1  :    ASA 3.2   | : |   ASA 4.3    : ASA 4.4    |
+ - - - - - - - - - - - - - + : + - - - - - - - - - - - - - +
             :                :                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|              Autonomic Networking Infrastructure          |
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
+--------+   :   +--------+   :   +--------+   :   +--------+
| Node 1 |-------| Node 2 |-------| Node 3 |-------| Node 4 |
+--------+   :   +--------+   :   +--------+   :   +--------+
```

Figure 1: High level view of an Autonomic Network

Figure 1 depicts 4 Nodes, 4 Autonomic Functions and 10 Autonomic Service Agents. Let’s list assumptions with regards of these elements.

Starting with nodes,

each may be either an Unconstrained Autonomic Node, a Constrained Autonomic Node (or even a legacy one?),

they may well be of different models (or having different software versions),

they may well be of different equipment vendors,

they may well be of different technologies (some may be IP routers, some may be Ethernet switches or OTN switches...).

Pursuing with Autonomic Functions,

they may well have different objectives (one could target automatic configuration of OSPF-TE, while another one is optimizing traffic load), but they may well have identical objectives as two could optimize energy consumption (possibly on different areas as function 3 and function 4),

each may be composed of no more than one ASA (either because the function is responsible for a single node or because the function relies on a centralized implementation),

each may well be composed of different sort of ASAs, meaning the software is different (either because their version number is different, or because the software provider is different, or because their respective nodes/equipments differ or because the role of each agent is different),

[Observation] Depending on the implementation the same piece of software may fulfill different roles or each role may come from a different from a different piece of code,

each has reached a given organization, meaning an organized set of ASAs in charge of a set of nodes ()whether formalized or not), this organization may either come from the piece of software itself (e.g. embedding a self-organization process) or come from directions of the network operator (e.g. through intents/policies, or through deployment instructions)

each may work internally in a peer to peer fashion (where every agents have the same prerogatives) or in hierarchical fashion (where some agents have some prerogatives over other) [this option is a good example of role differences],

each having its scope of work in terms of objective to reach and area/space/part of the network to manage.

Completing with individual Autonomic Service Agents, those are pieces of software:

embedded inside the node/equipment OS (hence present since the bootstrap or OS update of the equipment),

running in a machine different than the node (this could be a node controller or any other host or virtual machine)

[Observation] In the latter case, the ASA would likely require external credentials to interact with the node, directly monitoring and configuring the equipment (likely requires the ASA to be embedded) or through a management interface of the equipment (e.g. SNMP, TL1, Q3, NetConf) or through an equipment controller (e.g. SDN paradigm) or through a network manager (e.g. using the north interface of the manager)

either activated at start-up or as the result of a management action,

may be installed (either inside the equipment or on a different machine) when requested by an operator from a software origin (e.g. a repository in the ACP, a media)

provided by the same vendor as the equipment it manages or by any third party (like another equipment vendor, a management software vendor, an open-source initiative or the operator software team),

sharing a technical objective with the other ASAs of the Autonomic Function they belong, (or at least a similar one)?

can it contains multiple technical objective?

must the technical objective be intrinsic or can it be set by a managing entity (a network operator or a management system)?

The last three points being largely questionable are marked as questions.

The figure below illustrates how an ASA interacts with a node that the ASA manages. The left side depicts external interactions, through exchange of commands towards interfaces either to the node OS (e.g. via SNMP or NetConf), or to the controller (e.g. (G)MPLS, SDN, ...), or to the NMS. The right side depicts the case of the ASA embedded inside the Node OS.

```
   + - - - +       +-------------+
   |  ASA  |------>|     NMS     *<--*
   + - - - +       +------^------+   |
     |   |                |          |
     |   |         +------V------+   |
     |   +-------->| Controller  |   |
     |             +------^------+   |  +---------------------+
     |                    |          |  | + - - - +           |
     |             +------V------+   |  | |  ASA  |  Node OS  |
     +------------>|   Node OS   *<--*  | + - - - +           |
                   +------^------+      +--------------*------+
                          |                            |
                   +------V------+               +-----*------+
                   |     Node    |               |    Node    |
                   +-------------+               +------------+
```

Figure 2: Interaction possibilities between ASA and Resources

2.3. Operator’s requirements with regards to autonomic functions

Regarding the operators, at this point we can try to list few requirements they may have with regards with the Autonomic Functions and their management...

Being capable to set those functions a scope of work in term of area of duty,

Being capable to monitor the actions taken by the autonomic functions, and which are its results (performance with regards to the function KPIs)

Being capable to halt/suspend the execution of an Autonomic function (either because the function is untrusted, or because an operation on the network is to be conducted without interference from the autonomic functions, etc...)

Being capable to configure the autonomic functions by adjusting the parameters of the function (e.g. a Traffic Engineering autonomic function may achieve a trade-off between congestion avoidance and electrical power consumption, hence this function may be more or less aggressive on the link load ratio, and the network operator certainly has his word to say in setting this cursor.

3. 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.

As mentioned in the Problem statement section, an Autonomic Function Agent (ASA) controls resources of one or multiple Network Elements (NE), e.g. the interfaces of a router for a Load Balancing ASA. An ASA is a software, thus an ASA need first to be installed and to execute on a host machine in order to control resources.

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.

3.1. Operator's goal

In the installation phase, the operator's goal is to install ASA classes on Installation Hosts such that, at the moment of instantiation, the corresponding ASAs can control the sets of target resources. The complexity of the installation phase come from the number of possible configurations for the matching between the ASA classes capabilities (e.g. what types of resources it can control, what types of hosts it can be installed on...), the Installation Hosts capabilities (e.g. support dynamic installation, location and reachability...) and the operator's needs (what deployment schemes are favored, functionality coverage vs. cost trade-off...).

For example, in the coupled mode, the ASA host machine and the network element are the same. The ASA is installed on the network element and control the resources via interfaces and mechanisms internal to the network element. An ASA MUST be installed on the network element of every resource controlled by the ASA. The identification of the resources controlled by an ASA is straightforward: the resources are the ones of the network element.

In the decoupled mode, the ASA host machine is different from the network element. The ASA is installed on the host machine and control the resources via interfaces and mechanisms external to the network element. An ASA can be installed on an arbitrary set of candidate Installation hosts, which can be defined explicitly by the network operator or according to a cost function. A key benefit of the decoupled mode is to allow an easier introduction of autonomic functions on existing (legacy) infrastructure. The decoupled mode also allows de-correlating the installation requirements (compatible host machines) from the infrastructure evolution (NEs addition and removal, change of NE technology/version...).

The operator's goal may be defined as a special type of intent, called the Installation phase intent. The details of the content and format of this proposed intent are left open and for further study.

3.2. 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.

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).

4. 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).

4.1. 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]

4.2. 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,

4.3. 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. Instance Mandates are described in the requirements part of this document, which lists the needed fields of such a message (see Section 6.3 - ASA Instance Mandate).

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. Instance Manifests are described in the requirements part of this document, which lists the needed fields of such a message (see Section 6.4 - ASA Instance Manifest).

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:

- o the coordination entities (see [I-D.ciavaglia-anima-coordination] - Autonomic Functions Coordination)
- o 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) (see knowledge exchange items in Section 5 - Operation phase)

5. Operation phase

Note: This section is to be further developed in future revisions of the document.

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

6. Autonomic Function Agent specifications

6.1. Life-cycle

Based on the phases described above, this section defines formally the different states experienced by Autonomic Function Agents during their complete life-cycle.

The drawing of the life-cycle presented below shows both the states and the events/messages triggering the state changes. For simplification purposes, this sketch does not display the transitory states which correspond to the handling of the messages.

The installation and Instantiation phase will be concluded by ASA reaching respectively Installed and Instantiated states.

```
                    +--------------+
 Undeployed ------>|              |------> Undeployed
                    |  Installed   |
                +-->|              |---+
       Mandate  |   +--------------+   | Receives a
     is revoked |   +--------------+   |  Mandate
                +---|              |<--+
                    | Instantiated |
                +-->|              |---+
            set |   +--------------+   | set
           down |   +--------------+   | up
                +---|              |<--+
                    |  Operational |
                    |              |
                    +--------------+
```

Figure 3: Life cycle of an Autonomic Function Agent

Here are described the successive states of ASA.

Undeployed - In this "state", the Autonomic Function Agent is a mere piece of software, which may not even be available on any host.

Installed - In this state, the Autonomic Function Agent is available on a (/multiple) host(s), and after having shared its ASA class Manifest (which describes in a generic way independently of the deployment how the ASA would work). In this state the ASA is waiting for an ASA Instance Mandate, to determine which resources ti manage (when the ASA is strictly coupled to resources [e.g. part of a Node OS], there is no need to wait for an instance mandate, the target resources being intrinsically known).

Instantiated - In this state the Autonomic Function Agent has the knowledge of which resources it is meant to manage. In this state the ASA is expecting a set Up message in order to start executing its autonomic loop. From this state on the ASA can share an Instance Manifest (which describes how the ASA instance is going to work).

Operational - In this state, ASAs are executing their autonomic loop, hence acting on network, by modifying resources parameters. A set down message will turn back the ASA in an Instantiated state.

The messages are described in the following sections.

6.2. ASA Class Manifest

An ASA class is a piece of software that contains the computer program of an Autonomic Function Agent.

In order to install and instantiate appropriately an autonomic function in its network, the operator needs to know which are the characteristics of this piece of software.

This section details a format for an ASA class manifest, which is (a machine-readable) description of both the autonomic function and the piece of code that executes the function.

Field Name	Type	Description
ID	Struct	A unique identifier made out of at least a Function Name, Version and Provider Name (and Release Date).
Description	Struct	A multi-field description of the function performed by the ASA, it is meant to be read by the operator and can point to URLs, user-guides, feature descriptions.
Installation	3 Booleans	Whether the ASA is dynamically

properties		installable, can be decoupled from the NE and can manage multiple resources from a single instance (see Section 1 - Problem statement).
Possible Hosts	OS...	Lists the OS/Machines on which the ASA can be executed. [Only if ASA is dynamically installable]
Network Segment	NetSegment...	Lists the network segments on which the autonomic function is applicable (e.g. IP backbone versus RAN).
Manageable Equipments	Equipments...	Lists the nodes/resources that this piece of code can manage (e.g. ALU 77x routers, Cisco CRS-x routers, Huawei NEXE routers).
Autonomic Loop Type	Enum	States what is the type of loop MAPE-K and whether this loop can be halted in the course of its execution.
Acquired Inputs	Raw InfoSpec...	Lists the nature of information that an ASA agent may acquire from the managed resource(s) (e.g. the links load).
External Inputs	Raw InfoSpec...	Lists the nature of information that an ASA agent may require/wish from other ASA in the ecosystem that could provide such information/knowledge.
Possible Actions	Raw ActionSpec	Lists the nature of actions that an ASA agent may enforce on ASA the managed resource(s) (e.g. modify the links metrics).
Technical Objectives Description	Technical Objective Spec...	Lists the type of technical objectives that can be handled/received by the ASA (e.g. a max load of links).

Table 1: Fields of ASA class manifest

6.3. ASA Instance Mandate

An ASA instance is the ASA agent: a running piece of software of an ASA class. A software agent is a persistent, goal-oriented computer program that reacts to its environment and interacts with other elements of the network.

In order to install and instantiate appropriately an autonomic function in its network, the operator may specify to ASA instances what they are supposed to do: in term of which resources to manage and which objective to reach.

This section details a format for an ASA Instance Mandate, which is (a machine-readable) set of instructions sent to create autonomic functions out of ASA.

Field Name	Type	Description
ASA class Pattern	Struct	A pattern matching the ID (or part of it) of ASAs being the target of the Mandate. This field makes sense only for broadcast mandates (see end of this section).
Managed Resources	ResourcesId...	The list of resources to be managed by the ASA (e.g. their IP@ or MAC@ or any other relevant ID).
ID of Coord	Interface Id	The interface to the coordination system in charge of this autonomic function.
Reporting Policy	Policy	A policy describing which entities expect report from ASA, and which are the conditions of these reports (e.g. time wise and content wise)

Table 2: Fields of ASA instance mandate

An ASA instance mandate could be either:

sent to a targeted ASA In which case, the receiving Agent will have to manage the specified list of resources,

broadcast to all ASA In which case, the ASAs would collectively determine which agent would handle which resources from the list, and if needed (and feasible) this could also trigger the dynamic installation/instantiation of new agents (ACP should be capable of bearing such scenarios).

6.4. ASA Instance Manifest

Once the ASAs are properly instantiated, the operator and its managing system need to know which are the characteristics of these ASAs.

This section details a format for an ASA instance manifest, which is (a machine-readable) description of either an ASA or a set of ASAs gathered into an autonomic function.

Field Name	Type	Description
ASA Class ID	Struct	A unique identifier made out of at least a Function Name, Version and Provider Name (and Release Date).
ASA Instance ID	Long	A unique Id of the ASA instance (if the ASA instance manifest gathers multiple ASAs working together, this would be a list).
Hosts	Resource ID	ID of the Machines on which the ASA executes.
Managed Resources	ResourcesId...	The list of resources effectively managed by the ASA (e.g. their IP@ or MAC@ or any other relevant ID).
Acquired Inputs	Instance InfoSpec...	Lists information that this ASA agent may acquire from the managed resource(s) (e.g. the links load over links with ID x and y).
External Inputs	Instance InfoSpec...	Lists information that this ASA agent requires from the ecosystem (e.g. the links load prediction over links with ID x and y).
Possible Actions	Instance ActionSpec	Lists actions that this ASA agent may enforce on its managed resource(s) (e.g. modify the links metrics over links x and y).

Table 3: Fields of ASA instance manifest

7. Implication for other ANIMA components

7.1. Additional entities for ANIMA ecosystem

In the previous parts of this document, we have seen successive operations pertaining to the management of autonomic functions. These phases involve different entities such as the ASAs, the ASA Hosts and the ASA Management function. This function serves as the interface between the network operator and its managed infrastructure (i.e. the autonomic network). The ASA management function distributes instructions to the ASAs such as the ASA Instance Mandate, ASA set up/set down commands and also trigger the ASA installation inside ASA Hosts. This function is likely to be co-located or integrated with the function responsible for the management of the Intents.

In this first version, we do not prescribe any requirements on the way the ASA Management function should be implemented, neither the various deployment options of such a function and neither on the way ACP or GRASP could be extended to interact with this function. We believe these design and specifications work should be first discussed and analyzed by the working group.

7.2. Requirements for GRASP and ACP messages

GRASP and ACP seems to be the best (and currently only) candidates to convey the following messages between the ASA Management function and the ASAs:

ASA Class Manifest

ASA Instance Mandate (and Revoke Mandate)

ASA Instance Manifest

Set Up and Set Down messages

These section concludes with requests to GRASP protocol designers in order to handle the 3 last messages of the list above. These 3 messages form the minimal set of features needed to guarantee some control on the behavior of ASAs to network operators.

A mechanism similar to the bootstrapping one would usefully achieve discovery of pre-installed ASAs, and possibly provide those with a default Instance Mandate.

A mechanism to achieve dynamic installation of ASAs compatible with ACP and GRASP remains to be identified.

In the case of decoupled ASAs, even more for the ones supporting multiplicity, when a Mandate is broadcast (i.e. requesting the Instantiation of an autonomic function to manage a bunch of resources), these ASAs require synchronization to determine which agent(s) will manage which resources. Proper ACP/GRASP messages supporting such a mechanism have to be identified together with protocol authors.

7.2.1. Control when an ASA runs

To control when an ASA runs (and possibly how it runs), the operator needs the capacity to start and stop ASAs. That is why an imperative command type of message is requested from GRASP.

Additionally this type of message could also be used to specify how the ASA is meant to run, e.g. whether its control loop is subdued to some constraints in terms of pace of execution or rhythm of execution (once a second, once a minute, once a day...)

Below a suggestion for GRASP:

In fragmentary CDDL, an Imperative message follows the pattern:

```
imperative-message = [M_IMPERATIVE, session-id, initiator, objective]
```

...

7.2.2. Know what an ASA does to the network

To know what an ASA does to the network, the operator needs to have the information of the elements either monitored or modified by the ASA, hence this ASA should disclose those.

The disclosing should take the form of a ASA Instance Manifest (see Section 6.4 - ASA Instance Manifest), which could be conveyed inside a GRASP discovery message, hence the fields of the ASA Instance Manifest would be conveyed inside the objective.

At this stage there are two options:

The whole manifest is conveyed as an objective.

Each field of the manifest is conveyed as an individual objective, more precisely, the acquired inputs would appear as discovery only, and the modifiable parameters would appear as negotiation objective. The unclear part is the expression of requested fields (when the ASA claims being a client for such objective). Could one of the already existing objective options a good match or should a new one be created.

...

7.2.3. Decide which ASA control which equipment

To determine which ASA controls which equipment (or vice-versa which equipments are controlled by which ASAs), the operators needs to be able to instruct ASA before the end of their bootstrap procedure.

These instructions sent to ASA during bootstrapping should take the format of an ASA Instance Mandate (see Section 6.3 - ASA Instance Mandate). This ASA Instance Mandate are sorts of Intents, and as GRASP is meant to handle Intents in a near future, it would be beneficial to already identify which sort of GRASP message are meant to be used by Intent in order to already define those. An option could be to reuse the Imperative messages defined above.

...

8. Acknowledgments

This draft was written using the xml2rfc project.

This draft content builds upon work achieved during UniverSelf FP7 EU project.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

TBD

11. References

11.1. Normative References

[I-D.ciavaglia-anima-coordination]
Ciavaglia, L. and P. Peloso, "Autonomic Functions Coordination", draft-ciavaglia-anima-coordination-00 (work in progress), July 2015.

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

11.2. Informative References

[I-D.behringer-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Liu, B., Jeff, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-behringer-anima-reference-model-04 (work in progress), October 2015.

[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, <http://www.rfc-editor.org/info/rfc7575>.

Authors' Addresses

Peloso Pierre
Nokia
Villarceaux
Nozay 91460
FR

Email: pierre.peloso@nokia.com

Laurent Ciavaglia
Nokia
Villarceaux
Nozay 91460
FR

Email: laurent.ciavaglia@nokia.com