

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

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

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

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

This Internet-Draft will expire on September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	2
2. Problem Statement	4
3. Guiding principles	4
4. Initial sketch of a Coordination Function	6
4.1. Preliminary assumptions	6
4.2. Algorithms for coordination	7
4.3. Behavior of the coordination function	7
4.3.1. Times of the identification of interactions between AF	8
4.3.2. Times of the coordination of AF	9
4.4. Conclusions	9
5. External Requirements	10
5.1. Autonomic Function Descriptor (AFD)	10
5.2. control/command interface of AF	10
5.3. Interaction/Information Maps	11
6. Specifications	11
7. Acknowledgements	11
8. IANA Considerations	11
9. Security Considerations	11
10. References	11
10.1. Normative References	11
10.2. Informative References	12
Authors' Addresses	12

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).

- o Balance the autonomic network goal(s) and autonomic functions individual goal(s) (Congruence and Coherence properties).
- o 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).
- o Deviate the least possible autonomic functions from their design objective(s) and individual goal(s) (Liberality property).
- o 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).
- o Not impose any requirement on the internal specifications of autonomic functions (Independence property).
- o Support multiple coordination mechanism types (Plurality property).
- o Enable coordination mechanisms to be plugged in at deploy- and run-time (Modularity property).
- o 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).
- o 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).
- o 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).
- o Be able to operate equally well in a distributed or centralized manner (Distributivity property).
- o 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:

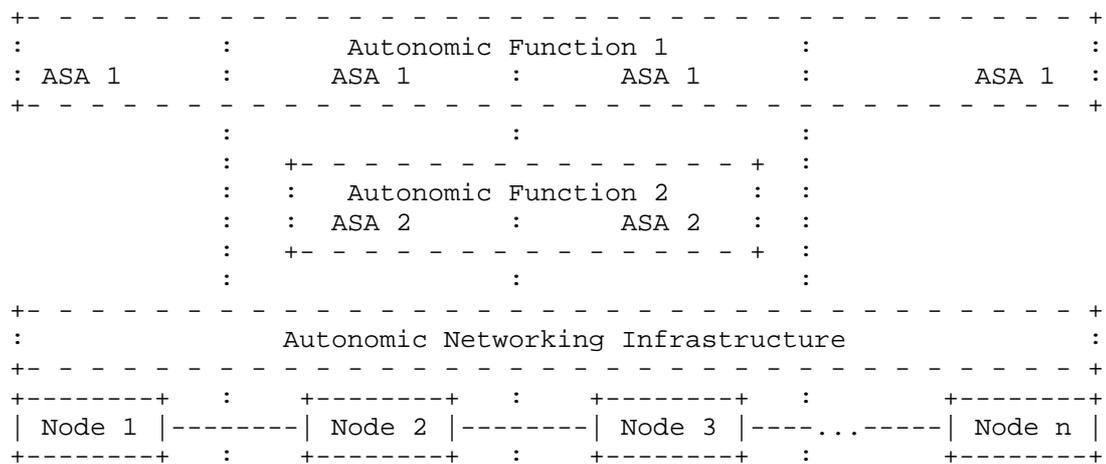


Figure 1: High level view of an Autonomic Network

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 row, 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 controlling 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
Internet-Draft
Intended status: Informational
Expires: August 18, 2017

Z. Du
S. Jiang
Huawei Technologies Co., Ltd
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

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

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

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

This Internet-Draft will expire on August 18, 2017.

Copyright Notice

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

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

Table of Contents

- 1. Introduction 2
- 2. Requirements Language and Terminology 3
- 3. Concept of ANIMA Intent Policy 4
- 4. Intent Life Cycle 4
- 5. Use Cases for ANIMA Intent Policy 6
- 6. Distribution of ANIMA Intent Policy 7
- 7. Management of ANIMA Intent Policy 7
- 8. Interpretation of ANIMA Intent Policy 7
- 9. Uniform Format of the ANIMA Intent Policy 8
- 10. Security Considerations 9
- 11. IANA Considerations 9
- 12. Acknowledgements 9
- 13. Change log [RFC Editor: Please remove] 9
- 14. References 9
- Authors' Addresses 10

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

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

S. Bjarnason
Arbor Networks
30 October 2020

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 3 May 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

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

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

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

Appendix B. Unfinished considerations (To Be Removed From RFC) 175

 B.1. Considerations for improving secure channel negotiation 175

 B.2. ACP address verification 176

 B.3. Public CA considerations 178

 B.4. Hardening DULL GRASP considerations 179

Authors' Addresses 179

1. Introduction (Informative)

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

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

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

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

Traditionally, physically separate, so-called out-of-band (management) networks have been used to avoid these problems or at least to allow recovery from such problems. Worst case, personnel are sent on site to access devices through out-of-band management ports (also called craft ports, serial console, management ethernet port). However, both options are expensive.

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

This document describes a modular design for a self-forming, self-managing and self-protecting ACP, which is a virtual out-of-band network designed to be as independent as possible of configuration, addressing and routing to avoid the self-dependency problems of current IP networks while still operating in-band on the same physical network that it is controlling and managing. The ACP design is therefore intended to combine as well as possible the resilience of out-of-band management networks with the low-cost of traditional IP in-band network management. The details how this is achieved are described in Section 6.

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

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

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

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

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

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

1. Autonomic functions communicate over the ACP. The ACP therefore directly supports Autonomic Networking functions, as described in [I-D.ietf-anima-reference-model]. For example, Generic Autonomic Signaling Protocol (GRASP - [I-D.ietf-anima-grasp]) runs securely inside the ACP and depends on the ACP as its "security and transport substrate".
2. A controller or network management system can use it to securely bootstrap network devices in remote locations, even if the (Data-Plane) network in between is not yet configured; no Data-Plane dependent bootstrap configuration is required. An example of such a secure bootstrap process is described in [I-D.ietf-anima-bootstrapping-keyinfra].
3. An operator can use it to access remote devices using protocols such as Secure SHell (SSH) or Network Configuration Protocol (NETCONF) running across the ACP, even if the network is misconfigured or not configured.

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

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

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

The ACP provides secure IPv6 connectivity, therefore it can be used not only as the secure connectivity for self-management as required for the ACP in [RFC7575], but it can also be used as the secure connectivity for traditional (centralized) management. The ACP can be implemented and operated without any other components of autonomic networks, except for the GRASP protocol. ACP relies on per-link DULL GRASP (see Section 6.4) to autodiscover ACP neighbors, and includes the ACP GRASP instance to provide service discovery for clients of the ACP (see Section 6.9) including for its own maintenance of ACP certificates.

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

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

1.1. Applicability and Scope

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

The design of the ACP as defined in this document is considered to be applicable to all types of "professionally managed" networks: Service Provider, Local Area Network (LAN), Metro(politan networks), Wide Area Network (WAN), Enterprise Information Technology (IT) and ->"Operational Technology" (OT) networks. The ACP can operate equally on layer 3 equipment and on layer 2 equipment such as bridges (see Section 7). The hop-by-hop authentication, integrity-protection and confidentiality mechanism used by the ACP is defined to be negotiable, therefore it can be extended to environments with different protocol preferences. The minimum implementation

requirements in this document attempt to achieve maximum interoperability by requiring support for multiple options depending on the type of device: IPsec, see [RFC4301], and Datagram Transport Layer Security (DTLS, see Section 6.8.4).

The implementation footprint of the ACP consists of Public Key Infrastructure (PKI) code for the ACP certificate including "Enrollment over Secure Transport (EST, see [RFC7030]), the GRASP protocol, UDP, TCP and Transport Layer Security (TLS, see Section 6.1), for security and reliability of GRASP and for EST, the ACP secure channel protocol used (such as IPsec or DTLS), and an instance of IPv6 packet forwarding and routing via the Routing Protocol for Low-power and Lossy Networks (RPL), see [RFC6550], that is separate from routing and forwarding for the Data-Plane (user traffic).

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

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

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

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

The ACP design can be applicable to devices constrained with respect to cpu and memory, and to networks constrained with respect to bitrate and reliability, but this document does not attempt to define the most constrained type of devices or networks to which the ACP is

applicable. RPL and DTLS for ACP secure channels are two protocol choices already making ACP more applicable to constrained environments. Support for constrained devices in this specification is opportunistic, but not complete, because the reliable transport for GRASP (see Section 6.9.2) only specifies TCP/TLS. See Appendix A.8 for discussions about how future standards or proprietary extensions/variations of the ACP could better meet different expectations from those on which the current design is based including supporting constrained devices better.

2. Acronyms and Terminology (Informative)

[RFC-Editor: Please add ACP, BRSKI, GRASP, MASA to <https://www.rfc-editor.org/materials/abbrev.expansion.txt>.]

[RFC-Editor: What is the recommended way to reference a hanging text, e.g. to a definition in the list of definitions? Up to -28, this document was using XMLv2 and the only option I could find for RFC/XML to point to a hanging text was `format="title"`, which leads to references such as `'->"ACP certificate" ()'`, aka: redundant empty parenthesis. Many reviewers were concerned about this. I created a ticket to ask for an xml2rfc enhancement to avoid this in the future: <https://trac.tools.ietf.org/tools/xml2rfc/trac/ticket/347>. When I changed to XMLv3 in version -29, I could get rid of the unnecessary `()` by using `format="none"`, but that format is declared to be deprecated in XMLv3. So I am not aware of any working AND "non-deprecated" option.]

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

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

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

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

- ACP: "Autonomic Control Plane". The Autonomic Function as defined in this document. It provides secure zero-touch (automated) transitive (network wide) IPv6 connectivity for all nodes in the same ACP domain as well as a GRASP instance running across this ACP IPv6 connectivity. The ACP is primarily meant to be used as a component of the ANI to enable Autonomic Networks but it can equally be used in simple ANI networks (with no other Autonomic Functions) or completely by itself.
- ACP address: An IPv6 address assigned to the ACP node. It is stored in the `acp-node-name` of the `->"ACP certificate"`.
- ACP address range/set: The ACP address may imply a range or set of addresses that the node can assign for different purposes. This address range/set is derived by the node from the format of the ACP address called the "addressing sub-scheme".
- ACP connect interface: An interface on an ACP node providing access to the ACP for non ACP capable nodes without using an ACP secure channel. See Section 8.1.1.
- ACP domain: The ACP domain is the set of nodes with `->"ACP certificates"` that allow them to authenticate each other as members of the ACP domain. See also Section 6.2.3.
- ACP (ANI/AN) certificate: A [RFC5280] certificate (LDevID) carrying the `acp-node-name` which is used by the ACP to learn its address in the ACP and to derive and cryptographically assert its membership in the ACP domain.
- ACP `acp-node-name` field: An information field in the ACP certificate in which the ACP relevant information is encoded: the ACP domain name, the ACP IPv6 address of the node and optional additional role attributes about the node.
- ACP Loopback interface: The Loopback interface in the ACP Virtual Routing and Forwarding (VRF) that has the ACP address assigned to it. See Section 6.13.5.1.
- ACP network: The ACP network constitutes all the nodes that have access to the ACP. It is the set of active and transitively connected nodes of an ACP domain plus all nodes that get access to the ACP of that domain via ACP edge nodes.
- ACP (ULA) prefix(es): The /48 IPv6 address prefixes used across the ACP. In the normal/simple case, the ACP has one ULA prefix, see Section 6.11. The ACP routing table may include multiple ULA prefixes if the "rsub" option is used to create addresses from more than one ULA prefix. See Section 6.2.2. The ACP may also include non-ULA prefixes if those are configured on ACP connect interfaces. See Section 8.1.1.
- ACP secure channel: A channel authenticated via `->"ACP certificates"`

providing integrity protection and confidentiality through encryption. These are established between (normally) adjacent ACP nodes to carry traffic of the ACP VRF securely and isolated from Data-Plane traffic in-band over the same link/path as the Data-Plane.

- ACP secure channel protocol: The protocol used to build an ACP secure channel, e.g., Internet Key Exchange Protocol version 2 (IKEv2) with IPsec or Datagram Transport Layer Security (DTLS).
- ACP virtual interface: An interface in the ACP VRF mapped to one or more ACP secure channels. See Section 6.13.5.
- AN "Autonomic Network": A network according to [I-D.ietf-anima-reference-model]. Its main components are ANI, Autonomic Functions and Intent.
- (AN) Domain Name: An FQDN (Fully Qualified Domain Name) in the acp-node-name of the Domain Certificate. See Section 6.2.2.
- ANI (nodes/network): "Autonomic Network Infrastructure". The ANI is the infrastructure to enable Autonomic Networks. It includes ACP, BRSKI and GRASP. Every Autonomic Network includes the ANI, but not every ANI network needs to include autonomic functions beyond the ANI (nor Intent). An ANI network without further autonomic functions can for example support secure zero-touch (automated) bootstrap and stable connectivity for SDN networks - see [RFC8368].
- ANIMA: "Autonomic Networking Integrated Model and Approach". ACP, BRSKI and GRASP are specifications of the IETF ANIMA working group.
- ASA: "Autonomic Service Agent". Autonomic software modules running on an ANI device. The components making up the ANI (BRSKI, ACP, GRASP) are also described as ASAs.
- Autonomic Function: A function/service in an Autonomic Network (AN) composed of one or more ASA across one or more ANI nodes.
- BRSKI: "Bootstrapping Remote Secure Key Infrastructures" ([I-D.ietf-anima-bootstrapping-keyinfra]. A protocol extending EST to enable secure zero-touch bootstrap in conjunction with ACP. ANI nodes use ACP, BRSKI and GRASP.
- CA: "Certification Authority". An entity that issues digital certificates. A CA uses its private key to sign the certificates it issues. Relying parties use the public key in the CA certificate to validate the signature.
- CRL: "Certificate Revocation List". A list of revoked certificates. Required to revoke certificates before their lifetime expires.
- Data-Plane: The counterpoint to the ACP VRF in an ACP node: forwarding of user traffic and in non-autonomous nodes/networks also any non-autonomous control and/or management plane functions. In a fully Autonomic Network node, the Data-Plane is managed autonomically via Autonomic Functions and Intent. See Section 1 for more detailed explanations.
- device: A physical system, or physical node.

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

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

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

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

in-band (management/signaling): In-band management traffic and/or control plane signaling uses the same network resources such as routers/switches and network links that it manages/controls. In-band is the standard management and signaling mechanism in IP networks. Compared to ->"out-of-band" it requires no additional physical resources, but introduces potentially circular dependencies for its correct operations. See ->"introduction".

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

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

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

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

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

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

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

NOC: Network Operations Center.

- node: A system supporting the ACP according to this document. Can be virtual or physical. Physical nodes are called devices.
- Node-ID: The identifier of an ACP node inside that ACP. It is the last 64 (see Section 6.11.3) or 78-bits (see Section 6.11.5) of the ACP address.
- OAM: Operations, Administration and Management. Includes Network Monitoring.
- Operational Technology (OT): https://en.wikipedia.org/wiki/Operational_Technology: "The hardware and software dedicated to detecting or causing changes in physical processes through direct monitoring and/or control of physical devices such as valves, pumps, etc.". OT networks are today in most cases well separated from Information Technology (IT) networks.
- out-of-band (management) network: An out-of-band network is a secondary network used to manage a primary network. The equipment of the primary network is connected to the out-of-band network via dedicated management ports on the primary network equipment. Serial (console) management ports were historically most common, higher end network equipment now also has ethernet ports dedicated only for management. An out-of-band network provides management access to the primary network independent of the configuration state of the primary network. See ->"Introduction"
- (virtual) out-of-band network: The ACP can be called a virtual out-of-band network for management and control because it attempts to provide the benefits of a (physical) ->"out-of-band network" even though it is physically carried ->"in-band". See ->"introduction".
- root CA: "root Certification Authority". A ->"CA" for which the root CA Key update procedures of [RFC7030], Section 4.4 can be applied.
- RPL: "IPv6 Routing Protocol for Low-Power and Lossy Networks". The routing protocol used in the ACP. See [RFC6550].
- (ACP/ANI/BRSKI) Registrar: An ACP registrar is an entity (software and/or person) that is orchestrating the enrollment of ACP nodes with the ACP certificate. ANI nodes use BRSKI, so ANI registrars are also called BRSKI registrars. For non-ANI ACP nodes, the registrar mechanisms are undefined by this document. See Section 6.11.7. Renewal and other maintenance (such as revocation) of ACP certificates may be performed by other entities than registrars. EST must be supported for ACP certificate renewal (see Section 6.2.5). BRSKI is an extension of EST, so ANI/BRSKI registrars can easily support ACP domain certificate renewal in addition to initial enrollment.
- RPI: "RPL Packet Information". Network extension headers for use with the ->"RPL" routing protocols. Not used with RPL in the ACP. See Section 6.12.1.13.
- RPL: "Routing Protocol for Low-Power and Lossy Networks". The routing protocol used in the ACP. See Section 6.12.

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

TA: "Trust Anchor". A Trust Anchor is an entity that is trusted for the purpose of certificate validation. Trust Anchor Information such as self-signed certificate(s) of the Trust Anchor is configured into the ACP node as part of Enrollment. See [RFC5280], Section 6.1.1.

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

ULA: (Global ID prefix) A "Unique Local Address" (ULA) is an IPv6 address in the block fc00::/7, defined in [RFC4193]. ULA is the IPv6 successor of the IPv4 private address space ([RFC1918]). ULA have important differences over IPv4 private addresses that are beneficial for and exploited by the ACP, such as the Locally Assigned Global ID prefix, which are the first 48-bits of a ULA address [RFC4193], section 3.2.1. In this document this prefix is abbreviated as "ULA prefix".

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

(ACP) Zone: An ACP zone is a set of ACP nodes using the same zone field value in their ACP address according to Section 6.11.3. Zones are a mechanism to support structured addressing of ACP addresses within the same /48-bit ULA prefix.

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

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

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

3.1. An Infrastructure for Autonomic Functions

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

3.2. Secure Bootstrap over a not configured Network

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

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

3.3. Data-Plane Independent Permanent Reachability

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

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

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

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

Assuming appropriate implementation (see Section 6.13.2 for more details), the ACP provides reachability that is independent of the Data-Plane. This allows the control plane and OAM plane to operate more robustly:

- * For management plane protocols, the ACP provides the functionality of a Virtual out-of-band (VooB) channel, by providing connectivity to all nodes regardless of their Data-Plane configuration, routing and forwarding tables.
- * For control plane protocols, the ACP allows their operation even when the Data-Plane is temporarily faulty, or during transitional events, such as routing changes, which may affect the control plane at least temporarily. This is specifically important for autonomic service agents, which could affect Data-Plane connectivity.

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

4. Requirements (Informative)

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

- ACP1: The ACP should provide robust connectivity: As far as possible, it should be independent of configured addressing, configuration and routing. Requirements 2 and 3 build on this requirement, but also have value on their own.
- ACP2: The ACP must have a separate address space from the Data-Plane. Reason: traceability, debug-ability, separation from Data-Plane, infrastructure security (filtering based on known address space).
- ACP3: The ACP must use autonomically managed address space. Reason: easy bootstrap and setup ("autonomic"); robustness (admin cannot break network easily). This document uses Unique Local Addresses (ULA) for this purpose, see [RFC4193].
- ACP4: The ACP must be generic, that is it must be usable by all the functions and protocols of the ANI. Clients of the ACP must not be tied to a particular application or transport protocol.
- ACP5: The ACP must provide security: Messages coming through the ACP must be authenticated to be from a trusted node, and it is very strongly > recommended that they be encrypted.

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

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

5. Overview (Informative)

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

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

Note:

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

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

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

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

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

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

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

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

6.2. ACP Domain, Certificate and Network

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

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

The LDevID certificate is called the ACP certificate. The TA is the Certification Authority (CA) root certificate of the ACP domain.

The ACP does not mandate specific mechanisms by which this keying material is provisioned into the ACP node. It only requires the certificate to comply with Section 6.2.1, specifically to have the acp-node-name as specified in Section 6.2.2 in its domain certificate as well as those of candidate ACP peers. See Appendix A.2 for more information about enrollment or provisioning options.

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

[RFC7575] defines the term "Autonomic Domain" as a collection of autonomic nodes. ACP nodes do not need to be fully autonomic, but when they are, then the ACP domain is an autonomic domain. Likewise, [I-D.ietf-anima-reference-model] defines the term "Domain Certificate" as the certificate used in an autonomic domain. The ACP

certificate is that domain certificate when ACP nodes are (fully) autonomic nodes. Finally, this document uses the term ACP network to refer to the network created by active ACP nodes in an ACP domain. The ACP network itself can extend beyond ACP nodes through the mechanisms described in Section 8.1.

6.2.1. ACP Certificates

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.2.2. ACP Certificate AcpNodeName

```

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

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

```

Example:

```

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

```

then this results in:

```

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

```

Figure 2: ACP Node Name ABNF

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

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

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

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

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

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

The following points explain and justify the encoding choices described:

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

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

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

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

6.2.2.1. AcpNodeName ASN.1 Module

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

```

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

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

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

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

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

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

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

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

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

END

```

Figure 3

6.2.3. ACP domain membership check

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

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

- 2: The peer's certificate passes certificate path validation as defined in [RFC5280], section 6 against one of the TA associated with the ACP node's ACP certificate (see Section 6.2.4 below). This includes verification of the validity (lifetime) of the certificates in the path.
- 3: If the peer's certificate indicates a Certificate Revocation List (CRL) Distribution Point (CRLDP) ([RFC5280], section 4.2.1.13) or Online Certificate Status Protocol (OCSP) responder ([RFC5280], section 4.2.2.1), then the peer's certificate MUST be valid according to those mechanisms when they are available: An OCSP check for the peer's certificate across the ACP must succeed or the peer certificate must not be listed in the CRL retrieved from the CRLDP. These mechanisms are not available when the ACP node has no ACP or non-ACP connectivity to retrieve a current CRL or access an OCSP responder and the security association protocol itself has also no way to communicate CRL or OCSP check. Retries to learn revocation via OCSP/CRL SHOULD be made using the same backoff as described in Section 6.7. If and when the ACP node then learns that an ACP peer's certificate is invalid for which rule 3 had to be skipped during ACP secure channel establishment, then the ACP secure channel to that peer MUST be closed even if this peer is the only connectivity to access CRL/OCSP. This applies (of course) to all ACP secure channels to this peer if there are multiple. The ACP secure channel connection MUST be retried periodically to support the case that the neighbor acquires a new, valid certificate.
- 4: The peer's certificate has a syntactically valid acp-node-name field and the acp-domain-name in that peer's acp-node-name is the same as in this ACP node's certificate (lowercase normalized).

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

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

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

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

The special value 0 in an ACP certificates acp-address field is used for nodes that can and should determine their ACP address through other mechanisms than learning it through the acp-address field in their ACP certificate. These ACP nodes are permitted to establish ACP secure channels. Mechanisms for those nodes to determine their ACP address are outside the scope of this specification, but this option is defined here so that any ACP nodes can build ACP secure channels to them according to Rule 5.

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

In summary:

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

For brevity, the remainder of this document refers to this process only as authentication instead of as authentication and authorization.

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

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

6.2.3.1. Realtime clock and Time Validation

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

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

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

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

6.2.4. Trust Anchors (TA)

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

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

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

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

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

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

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

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

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

6.2.5. Certificate and Trust Anchor Maintenance

ACP nodes MUST support renewal of their Certificate and TA information via EST and MAY support other mechanisms. See Section 6.1 for TLS requirements. An ACP network MUST have at least one ACP node supporting EST server functionality across the ACP so that EST renewal is useable.

ACP nodes SHOULD be able to remember the IPv6 locator parameters of the O_IPv6_LOCATOR in GRASP of the EST server from which they last renewed their ACP certificate. They SHOULD provide the ability for

these EST server parameters to also be set by the ACP Registrar (see Section 6.11.7) that initially enrolled the ACP device with its ACP certificate. When BRSKI (see [I-D.ietf-anima-bootstrapping-keyinfra]) is used, the IPv6 locator of the BRSKI registrar from the BRSKI TLS connection SHOULD be remembered and used for the next renewal via EST if that registrar also announces itself as an EST server via GRASP (see next section) on its ACP address.

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

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

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

6.2.5.1. GRASP objective for EST server

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

Example:

```
[M_FLOOD, 12340815, h'fd89b714f3db0000200000064000001', 210000,
  [{"SRV.est", 4, 255 }],
  [O_IPv6_LOCATOR,
    h'fd89b714f3db0000200000064000001', IPPROTO_TCP, 443]]
]
```

Figure 4: GRASP SRV.est example

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

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

objective = ["SRV.est", objective-flags, loop-count,
            objective-value]

objective-flags = sync-only ; as in GRASP spec
sync-only      = 4         ; M_FLOOD only requires synchronization
loop-count     = 255      ; recommended as there is no mechanism
                  ; to discover network diameter.
objective-value = any     ; reserved for future extensions
```

Figure 5: GRASP SRV.est definition

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

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

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

6.2.5.2. Renewal

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

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

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

6.2.5.3. Certificate Revocation Lists (CRLs)

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

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

A CRLDP can be reachable across the ACP either by running it on a node with ACP or by connecting its node via an ACP connect interface (see Section 8.1).

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

6.2.5.4. Lifetimes

Certificate lifetime may be set to shorter lifetimes than customary (1 year) because certificate renewal is fully automated via ACP and EST. The primary limiting factor for shorter certificate lifetimes is load on the EST server(s) and CA. It is therefore recommended that ACP certificates are managed via a CA chain where the assigning CA has enough performance to manage short lived certificates. See also Section 9.2.4 for discussion about an example setup achieving this. See also [I-D.ietf-acme-star].

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

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

6.2.5.5. Re-enrollment

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

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

Please refer to Section 6.11.7 and [I-D.ietf-anima-bootstrapping-keyinfra] for explanations about ACP Registrars and vouchers as used in the following text. When ACP is intended to be used without BRSKI, the details about BRSKI and vouchers in the following text can be skipped.

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

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

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

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

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

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

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

6.2.5.6. Failing Certificates

An ACP certificate is called failing in this document, if/when the ACP node to which the certificate was issued can determine that it was revoked (or explicitly not renewed), or in the absence of such explicit local diagnostics, when the ACP node fails to connect to other ACP nodes in the same ACP domain using its ACP certificate. For connection failures to determine the ACP certificate as the culprit, the peer should pass the domain membership check (Section 6.2.3) and other reasons for the connection failure can be excluded because of the connection error diagnostics.

This type of failure can happen during setup/refresh of a secure ACP channel connections or any other use of the ACP certificate, such as for the TLS connection to an EST server for the renewal of the ACP domain certificate.

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

ACP nodes SHOULD support the option to determine whether its ACP certificate is failing, and when it does, put itself into the role of a re-enrolling candidate ACP node as explained above (Section 6.2.5.5).

6.3. ACP Adjacency Table

To know to which nodes to establish an ACP channel, every ACP node maintains an adjacency table. The adjacency table contains information about adjacent ACP nodes, at a minimum: Node-ID (identifier of the node inside the ACP, see Section 6.11.3 and Section 6.11.5), interface on which neighbor was discovered (by GRASP as explained below), link-local IPv6 address of neighbor on that interface, certificate (including acp-node-name). An ACP node MUST maintain this adjacency table. This table is used to determine to which neighbor an ACP connection is established.

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

The adjacency table MAY contain information about the validity and trust of the adjacent ACP node's certificate. However, subsequent steps MUST always start with the ACP domain membership check against the peer (see Section 6.2.3).

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

6.4. Neighbor Discovery with DULL GRASP

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

Discovery Unsolicited Link-Local (DULL) GRASP is a limited subset of GRASP intended to operate across an insecure link-local scope. See section 2.5.2 of [I-D.ietf-anima-grasp] for its formal definition. The ACP uses one instance of DULL GRASP for every L2 interface of the ACP node to discover link level adjacent candidate ACP neighbors. Unless modified by policy as noted earlier (Section 5 bullet point 2.), native interfaces (e.g., physical interfaces on physical nodes) SHOULD be initialized automatically to a state in which ACP discovery can be performed and any native interfaces with ACP neighbors can then be brought into the ACP even if the interface is otherwise not configured. Reception of packets on such otherwise not configured interfaces MUST be limited so that at first only IPv6 Stateless Address Auto Configuration (SLAAC - [RFC4862]) and DULL GRASP work

and then only the following ACP secure channel setup packets - but not any other unnecessary traffic (e.g., no other link-local IPv6 transport stack responders for example).

Note that the use of the IPv6 link-local multicast address (ALL_GRASP_NEIGHBORS) implies the need to use Multicast Listener Discovery Version 2 (MLDv2, see [RFC3810]) to announce the desire to receive packets for that address. Otherwise DULL GRASP could fail to operate correctly in the presence of MLD snooping ([RFC4541]) switches that are not ACP supporting/enabled - because those switches would stop forwarding DULL GRASP packets. Switches not supporting MLD snooping simply need to operate as pure L2 bridges for IPv6 multicast packets for DULL GRASP to work.

ACP discovery SHOULD NOT be enabled by default on non-native interfaces. In particular, ACP discovery MUST NOT run inside the ACP across ACP virtual interfaces. See Section 9.3 for further, non-normative suggestions on how to enable/disable ACP at node and interface level. See Section 8.2.2 for more details about tunnels (typical non-native interfaces). See Section 7 for how ACP should be extended on devices operating (also) as L2 bridges.

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

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

```
[M_FLOOD, 12340815, h'fe80000000000000c0011001feef0000', 210000,
  [{"AN_ACP", 4, 1, "IKEv2" },
   [O_IPv6_LOCATOR,
    h'fe80000000000000c0011001feef0000', IPPROTO_UDP, 15000]]
 [{"AN_ACP", 4, 1, "DTLS" },
  [O_IPv6_LOCATOR,
   h'fe80000000000000c0011001feef0000', IPPROTO_UDP, 17000]]
]
```

Figure 6: GRASP AN_ACP example

The formal CDDL definition is:

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

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

objective-flags = sync-only ; as in the GRASP specification
sync-only = 4 ; M_FLOOD only requires synchronization
loop-count = 1 ; limit to link-local operation

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

```

Figure 7: GRASP AN_ACP definition

The objective-flags field is set to indicate synchronization.

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

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

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

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

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

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

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

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

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

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

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

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

A node that supports more than one secure channel protocol method needs to flood multiple versions of the "AN_ACP" objective so that each method can be accompanied by its own locator-option. This can use a single GRASP M_FLOOD message as shown in Figure 6.

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

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

The result of the discovery is the IPv6 link-local address of the neighbor as well as its supported secure channel protocols (and non-standard port they are running on). It is stored in the ACP Adjacency Table (see Section 6.3), which then drives the further building of the ACP to that neighbor.

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

6.5. Candidate ACP Neighbor Selection

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

The ACP is established exclusively between nodes in the same domain. This includes all routing subdomains. Appendix A.6 explains how ACP connections across multiple routing subdomains are special.

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

6.6. Channel Selection

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

From the use-cases it seems clear that not all type of ACP nodes can or need to connect directly to each other or are able to support or prefer all possible mechanisms. For example, code space limited IoT

devices may only support DTLS because that code exists already on them for end-to-end security, but low-end in-ceiling L2 switches may only want to support Media Access Control Security (MacSec, see 802.1AE ([MACSEC])) because that is also supported in their chips. Only a flexible gateway device may need to support both of these mechanisms and potentially more. Note that MacSec is not required by any profiles of the ACP in this specification. Instead, MacSec is mentioned as a likely next interesting secure channel protocol. Note also that the security model allows and requires for any-to-any authentication and authorization between all ACP nodes because there is also end-to-end and not only hop-by-hop authentication for secure channels.

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

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

- * An ACP node may choose to attempt to initiate the different feasible ACP secure channel protocols it supports according to its local policies sequentially or in parallel, but it MUST support acting as a responder to all of them in parallel.
- * Once the first ACP secure channel protocol connection to a specific peer IPv6 address passes peer authentication, the two peers know each other's certificate because those ACP certificates are used by all secure channel protocols for mutual authentication. The peer with the higher Node-ID in the AcpNodeName of its ACP certificate takes on the role of the Decider towards the peer. The other peer takes on the role of the Follower. The Decider selects which secure channel protocol to ultimately use.
- * The Follower becomes passive: it does not attempt to further initiate ACP secure channel protocol connections with the Decider and does not consider it to be an error when the Decider closes secure channels. The Decider becomes the active party, continues to attempt setting up secure channel protocols with the Follower. This process terminates when the Decider arrives at the "best" ACP secure channel connection option that also works with the Follower ("best" from the Deciders point of view).
- * A peer with a "0" acp-address in its AcpNodeName takes on the role of Follower when peering with a node that has a non-"0" acp-address (note that this specification does not fully define the

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

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

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

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

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

Figure 8: Secure Channel sequence of steps

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

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

6.7. Candidate ACP Neighbor verification

Independent of the security association protocol chosen, candidate ACP neighbors need to be authenticated based on their domain certificate. This implies that any secure channel protocol MUST support certificate based authentication that can support the ACP domain membership check as defined in Section 6.2.3. If it fails, the connection attempt is aborted and an error logged. Attempts to reconnect MUST be throttled. The RECOMMENDED default is exponential base 2 backoff with an initial retransmission time (IRT) of 10 seconds and a maximum retransmission time (MRT) of 640 seconds.

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

6.8. Security Association (Secure Channel) protocols

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

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

6.8.1. General considerations

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

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

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

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

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

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

6.8.2. Common requirements

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

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

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

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

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

An ACP secure channel MUST immediately be terminated when the lifetime of any certificate in the chain used to authenticate the neighbor expires or becomes revoked. This may not be standard behavior in secure channel protocols because the certificate

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

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

6.8.3. ACP via IPsec

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

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

6.8.3.1. Native IPsec

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

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

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

IPsec tunnel mode is required because the ACP will route/forward packets received from any other ACP node across the ACP secure channels, and not only its own generated ACP packets. With IPsec transport mode (and no additional encapsulation header in the ESP payload), it would only be possible to send packets originated by the ACP node itself because the IPv6 addresses of the ESP must be the same as that of the outer IPv6 header.

6.8.3.1.1. RFC8221 (IPsec/ESP)

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

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

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

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

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

Explanations:

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

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

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

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

6.8.3.1.2. RFC8247 (IKEv2)

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

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

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

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

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

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

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

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

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

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

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

6.8.3.2. IPsec with GRE encapsulation

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

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

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

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

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

6.8.4. ACP via DTLS

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

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

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

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

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

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

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

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

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

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

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

6.8.5. ACP Secure Channel Profiles

As explained in the beginning of Section 6.6, there is no single secure channel mechanism mandated for all ACP nodes. Instead, this section defines two ACP profiles (baseline and constrained) for ACP nodes that do introduce such requirements.

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

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

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

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

6.9. GRASP in the ACP

6.9.1. GRASP as a core service of the ACP

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

ACP provides IP unicast routing via the RPL routing protocol (see Section 6.12).

The ACP does not use IP multicast routing nor does it provide generic IP multicast services (the handling of GRASP link-local multicast messages is explained in Section 6.9.2). Instead, the ACP provides service discovery via the objective discovery/announcement and negotiation mechanisms of the ACP GRASP instance (services are a form of objectives). These mechanisms use hop-by-hop reliable flooding of GRASP messages for both service discovery (GRASP M_DISCOVERY messages) and service announcement (GRASP M_FLOOD messages).

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

6.9.2. ACP as the Security and Transport substrate for GRASP

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

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

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

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

GRASP unicast messages inside the ACP always use the ACP address. Link-local addresses from the ACP VRF MUST NOT be used inside objectives. GRASP unicast messages inside the ACP are transported via TLS. See Section 6.1 for TLS requirements. TLS mutual authentication MUST use the ACP domain membership check defined in (Section 6.2.3).

GRASP link-local multicast messages are targeted for a specific ACP virtual interface (as defined Section 6.13.5) but are sent by the ACP into an ACP GRASP virtual interface that is constructed from the TCP connection(s) to the IPv6 link-local neighbor address(es) on the underlying ACP virtual interface. If the ACP GRASP virtual interface has two or more neighbors, the GRASP link-local multicast messages are replicated to all neighbor TCP connections.

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

6.9.2.1. Discussion

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

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

If GRASP peer connections were to use just TCP, compromised ACP members could simply eavesdrop passively on GRASP peer connections for whom they are on-path ("man in the middle" - MITM) or intercept and modify them. With TLS, it is not possible to completely eliminate problems with compromised ACP members, but attacks are a lot more complex:

Eavesdropping/spoofing by a compromised ACP node is still possible because in the model of the ACP and GRASP, the provider and consumer of an objective have initially no unique information (such as an identity) about the other side which would allow them to distinguish a benevolent from a compromised peer. The compromised ACP node would simply announce the objective as well, potentially filter the original objective in GRASP when it is a MITM and act as an application level proxy. This of course requires that the compromised ACP node understand the semantics of the GRASP negotiation to an extent that allows it to proxy it without being detected, but in an ACP environment this is quite likely public knowledge or even standardized.

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

- * Secure channel methods such as IPsec may provide protection against additional attacks, for example reset-attacks.
- * The secure channel method may leverage hardware acceleration and there may be little or no gain in eliminating it.

- * There is no different security model for ACP GRASP from other ACP traffic. Instead, there is just another layer of protection against certain attacks from the inside which is important due to the role of GRASP in the ACP.

6.10. Context Separation

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

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

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

6.11. Addressing inside the ACP

The channels explained above typically only establish communication between two adjacent nodes. In order for communication to happen across multiple hops, the autonomic control plane requires ACP network wide valid addresses and routing. Each ACP node creates a Loopback interface with an ACP network wide unique address (prefix) inside the ACP context (as explained in in Section 6.10). This address may be used also in other virtual contexts.

With the algorithm introduced here, all ACP nodes in the same routing subdomain have the same /48 ULA prefix. Conversely, ULA global IDs from different domains are unlikely to clash, such that two ACP networks can be merged, as long as the policy allows that merge. See also Section 10.1 for a discussion on merging domains.

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

6.11.1. Fundamental Concepts of Autonomic Addressing

- * Usage: Autonomic addresses are exclusively used for self-management functions inside a trusted domain. They are not used for user traffic. Communications with entities outside the

trusted domain use another address space, for example normally managed routable address space (called "Data-Plane" in this document).

- * Separation: Autonomic address space is used separately from user address space and other address realms. This supports the robustness requirement.
- * Loopback-only: Only ACP Loopback interfaces (and potentially those configured for "ACP connect", see Section 8.1) carry routable address(es); all other interfaces (called ACP virtual interfaces) only use IPv6 link local addresses. The usage of IPv6 link local addressing is discussed in [RFC7404].
- * Use-ULA: For Loopback interfaces of ACP nodes, we use ULA with L=1 (as defined in section 3.1 of [RFC4193]). Note that the random hash for ACP Loopback addresses uses the definition in Section 6.11.2 and not the one of [RFC4193] section 3.2.2.
- * No external connectivity: They do not provide access to the Internet. If a node requires further reaching connectivity, it should use another, traditionally managed address scheme in parallel.
- * Addresses in the ACP are permanent, and do not support temporary addresses as defined in [RFC4941].
- * Addresses in the ACP are not considered sensitive on privacy grounds because ACP nodes are not expected to be end-user hosts and ACP addresses do therefore not represent end-users or groups of end-users. All ACP nodes are in one (potentially federated) administrative domain. They are assumed to be to be candidate hosts of ACP traffic amongst each other or transit thereof. There are no transit nodes less privileged to know about the identity of other hosts in the ACP. Therefore, ACP addresses do not need to be pseudo-random as discussed in [RFC7721]. Because they are not propagated to untrusted (non ACP) nodes and stay within a domain (of trust), we also consider them not to be subject to scanning attacks.

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

- * Simplicity, reliability and scale: If other network layer protocols were supported, each would have to have its own set of security associations, routing table and process, etc.
- * Autonomic functions do not require IPv4: Autonomic functions and autonomic service agents are new concepts. They can be exclusively built on IPv6 from day one. There is no need for backward compatibility.
- * OAM protocols do not require IPv4: The ACP may carry OAM protocols. All relevant protocols (SNMP, TFTP, SSH, SCP, RADIUS, Diameter, NETCONF ...) are available in IPv6. See also [RFC8368] for how ACP could be made to interoperate with IPv4 only OAM.

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

6.11.2. The ACP Addressing Base Scheme

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

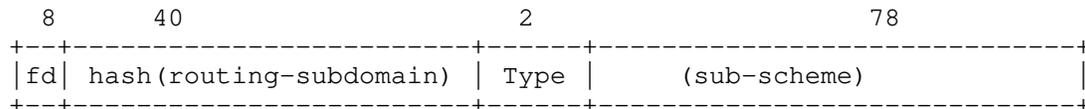


Figure 10: ACP Addressing Base Scheme

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

- * "fd" identifies a locally defined ULA address.
- * The 40-bits ULA "global ID" (term from [RFC4193]) for ACP addresses carried in the acp-node-name in the ACP certificates are the first 40-bits of the SHA256 hash of the routing subdomain from the same acp-node-name. In the example of Section 6.2.2, the routing subdomain is "area51.research.acp.example.com" and the 40-bits ULA "global ID" 89b714f3db.
- * When creating a new routing-subdomain for an existing autonomic network, it MUST be ensured, that rsub is selected so the resulting hash of the routing-subdomain does not collide with the hash of any pre-existing routing-subdomains of the autonomic network. This ensures that ACP addresses created by registrars for different routing subdomains do not collide with each other.
- * To allow for extensibility, the fact that the ULA "global ID" is a hash of the routing subdomain SHOULD NOT be assumed by any ACP node during normal operations. The hash function is only executed during the creation of the certificate. If BRSKI is used, then the BRSKI registrar will create the acp-node-name in response to the EST Certificate Signing Request (CSR) Attribute Request message by the pledge.

- * Establishing connectivity between different ACP (different acp-domain-name) is outside the scope of this specification. If it is being done through future extensions, then the rsub of all routing-subdomains across those autonomic networks need to be selected so the resulting routing-subdomain hashes do not collide. For example, a large cooperation with its own private TA may want to create different autonomic networks that initially should not be able to connect but where the option to do so should be kept open. When taking this future possibility into account, it is easy to always select rsub so that no collisions happen.
- * Type: This field allows different address sub-schemes. This addresses the "upgradability" requirement. Assignment of types for this field will be maintained by IANA.

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

Please refer to Section 6.11.7 and Appendix A.1 for further explanations why the following Sub-Addressing schemes are used and why multiple are necessary.

The following summarizes the addressing Sub-Schemes:

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

Figure 11: Addressing Sub-Schemes

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

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

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

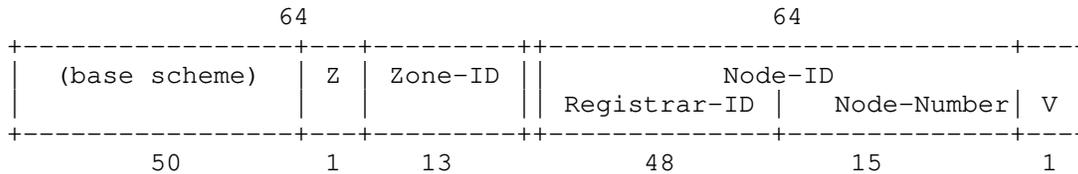


Figure 12: ACP Zone Addressing Sub-Scheme

The fields are defined as follows:

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

The 64-bit Node-ID must be unique across the ACP domain for each node. It is derived and composed as follows:

- * Registrar-ID (48-bit): A number unique inside the domain that identifies the ACP registrar which assigned the Node-ID to the node. One or more domain-wide unique identifiers of the ACP registrar can be used for this purpose. See Section 6.11.7.2.
- * Node-Number: Number to make the Node-ID unique. This can be sequentially assigned by the ACP Registrar owning the Registrar-ID.
- * V (1-bit): Virtualization bit: 0: Indicates the ACP itself ("ACP node base system"); 1: Indicates the optional "host" context on the ACP node (see below).

In the ACP Zone Addressing Sub-Scheme, the ACP address in the certificate has V field as all zero bits.

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

The Virtual bit in this sub-scheme allows the easy addition of the ACP as a component to existing systems without causing problems in the port number space between the services in the ACP and the existing system. V:0 is the ACP router (autonomic node base system), V:1 is the host with pre-existing transport endpoints on it that

could collide with the transport endpoints used by the ACP router. The ACP host could for example have a p2p virtual interface with the V:0 address as its router into the ACP. Depending on the software design of ASAs, which is outside the scope of this specification, they may use the V:0 or V:1 address.

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

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

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

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

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

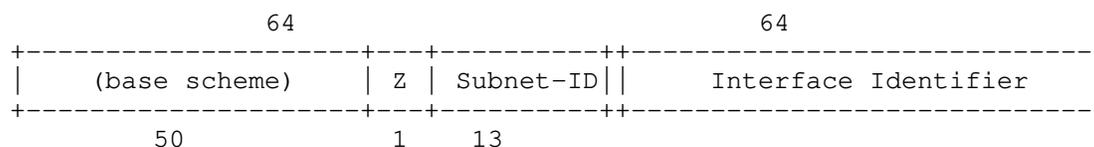


Figure 13: ACP Manual Addressing Sub-Scheme

The fields are defined as follows:

- * Type: MUST be 0x0.
- * Z: MUST be 0x1.
- * Subnet-ID: Configured subnet identifier.
- * Interface Identifier.

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

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

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

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

Address management of ACP connect subnets is done using traditional assignment methods and existing IPv6 protocols. See Section 8.1.3 for details. Therefore, the notion of V-bit many addresses assigned to the ACP nodes does not apply to this Sub-Scheme.

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

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

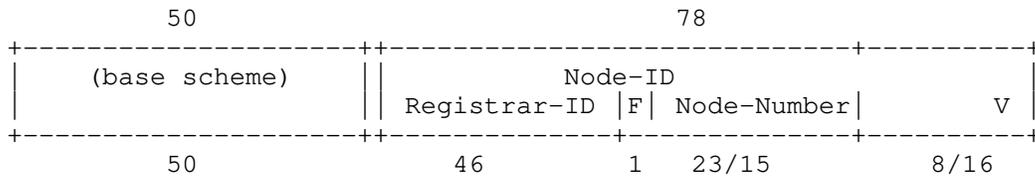


Figure 14: ACP Vlong Addressing Sub-Scheme

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

The fields are the same as in the Zone-ID sub-scheme with the following refinements:

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

The F=0 bit format addresses are intended to be used for "general purpose" ACP nodes that would potentially have a limited number (< 256) of clients (ASA/Autonomic Functions or legacy services) of the ACP that require separate V(irtual) addresses.

The F=1 bit Node-Numbers are intended for ACP nodes that are ACP edge nodes (see Section 8.1.1) or that have a large number of clients requiring separate V(irtual) addresses. For example, large SDN controllers with container modular software architecture (see Section 8.1.2).

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

6.11.6. Other ACP Addressing Sub-Schemes

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

6.11.7. ACP Registrars

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

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

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

6.11.7.1. Use of BRSKI or other Mechanism/Protocols

Any protocols or mechanisms may be used by ACP registrars, as long as the resulting ACP certificate and TA certificate(s) allow to perform the ACP domain membership described in Section 6.2.3 with other ACP domain members, and meet the ACP addressing requirements for its `acp-node-name` as described further below in this section.

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

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

6.11.7.2. Unique Address/Prefix allocation

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

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

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

When the candidate ACP device (called Pledge in BRSKI) is to be enrolled into an ACP domain, the ACP registrar needs to allocate a unique ACP address to the node and ensure that the ACP certificate gets a `acp-node-name` field (Section 6.2.2) with the appropriate information - ACP domain-name, ACP-address, and so on. If the ACP registrar uses BRSKI, it signals the ACP `acp-node-name` field to the Pledge via the `EST /csrattrs` command (see [I-D.ietf-anima-bootstrapping-keyinfra], section 5.9.2 - "EST CSR Attributes").

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

6.11.7.3. Addressing Sub-Scheme Policies

The ACP registrar selects for the candidate ACP node a unique address prefix from an appropriate ACP addressing sub-scheme, either a zone addressing sub-scheme prefix (see Section 6.11.3), or a Vlong addressing sub-scheme prefix (see Section 6.11.5). The assigned ACP address prefix encoded in the `acp-node-name` field of the ACP certificate indicates to the ACP node its ACP address information.

The sub-addressing scheme indicates the prefix length: /127 for zone address sub-scheme, /120 or /112 for Vlong address sub-scheme. The first address of the prefix is the ACP address. All other addresses in the prefix are for other uses by the ACP node as described in the zone and Vlong addressing sub scheme sections. The ACP address prefix itself is then signaled by the ACP node into the ACP routing protocol (see Section 6.12) to establish IPv6 reachability across the ACP.

The choice of addressing sub-scheme and prefix-length in the Vlong address sub-scheme is subject to ACP registrar policy. It could be an ACP domain wide policy, or a per ACP node or per ACP node type policy. For example, in BRSKI, the ACP registrar is aware of the IDevID certificate of the candidate ACP node, which typically contains a "serialNumber" attribute in the subject field distinguished name encoding that is often indicating the node's vendor and device type and can be used to drive a policy selecting an appropriate addressing sub-scheme for the (class of) node(s).

ACP registrars SHOULD default to allocate ACP zone sub-address scheme addresses with Zone-ID 0.

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

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

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

6.11.7.4. Address/Prefix Persistence

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

ACP address information SHOULD be maintained even when the renewing/rekeying ACP registrar is not the same as the one that enrolled the prior ACP certificate. See Section 9.2.4 for an example.

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

6.11.7.5. Further Details

Section 9.2 discusses further informative details of ACP registrars: What interactions registrars need, what parameters they require, certificate renewal and limitations, use of sub-CAs on registrars and centralized policy control.

6.12. Routing in the ACP

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

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

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

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

RPL adjacencies are set up across all ACP channels in the same domain including all its routing subdomains. See Appendix A.6 for more details.

6.12.1. ACP RPL Profile

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

6.12.1.1. Overview

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

6.12.1.1.1. Single Instance

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

This choice is a compromise. Consider a network that has multiple NOCs in different locations. Only one NOC will become the DODAG root. Traffic to and from other NOCs has to be sent through the DODAG (shortest path tree) rooted in the primary NOC. Depending on topology, this can be an annoyance from a latency point of view or from minimizing network path resources, but this is deemed to be acceptable given how ACP traffic is "only" network management/control traffic. See Appendix A.9.4 for more details.

Using a single instanceID/DODAG does not introduce a single point of failure, as the DODAG will reconfigure itself when it detects Data-Plane forwarding failures including choosing a different root when the primary one fails.

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

6.12.1.1.2. Reconvergence

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

The ACP RPL profile instead relies on quick reconverging the DODAG by recognizing link state change (down/up) and triggering reconvergence signaling as described in Section 6.12.1.7. Since links in the ACP are assumed to be mostly reliable (or have link layer protection against loss) and because there is no stretch according to Section 6.12.1.7, loops caused by loss of RPL routing protocol signaling packets should be exceedingly rare.

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

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

6.12.1.2. RPL Instances

Single RPL instance. Default RPLInstanceID = 0.

6.12.1.3. Storing vs. Non-Storing Mode

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

6.12.1.4. DAO Policy

Proactive, aggressive DAO state maintenance:

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

6.12.1.5. Path Metric

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

6.12.1.6. Objective Function

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

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

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

6.12.1.7. DODAG Repair

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

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

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

stretch_rank: none provided ("not stretched").

Data Path Validation: Not used.

Trickle: Not used.

6.12.1.8. Multicast

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

6.12.1.9. Security

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

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

6.12.1.10. P2P communications

Not used.

6.12.1.11. IPv6 address configuration

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

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

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

6.12.1.12. Administrative parameters

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

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

6.12.1.13. RPL Packet Information

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

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

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

6.12.1.14. Unknown Destinations

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

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

6.13. General ACP Considerations

Since channels are by default established between adjacent neighbors, the resulting overlay network does hop-by-hop encryption. Each node decrypts incoming traffic from the ACP, and encrypts outgoing traffic to its neighbors in the ACP. Routing is discussed in Section 6.12.

6.13.1. Performance

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

6.13.2. Addressing of Secure Channels

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

To avoid that Data-Plane configuration can impact the operations of the IPv6 (link-local) interface/address used for ACP channels, appropriate implementation considerations are required. If the IPv6

interface/link-local address is shared with the Data-Plane, it needs to be impossible to unconfigure/disable it through configuration. Instead of sharing the IPv6 interface/link-local address, a separate (virtual) interface with a separate IPv6 link-local address can be used. For example, the ACP interface could be run over a separate MAC address of an underlying L2 (Ethernet) interface. For more details and options, see Appendix A.9.2.

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

6.13.3. MTU

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

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

6.13.4. Multiple links between nodes

If two nodes are connected via several links, the ACP SHOULD be established across every link, but it is possible to establish the ACP only on a sub-set of links. Having an ACP channel on every link has a number of advantages, for example it allows for a faster failover in case of link failure, and it reflects the physical topology more closely. Using a subset of links (for example, a single link), reduces resource consumption on the node, because state needs to be kept per ACP channel. The negotiation scheme explained in Section 6.6 allows the Decider (the node with the higher ACP address) to drop all but the desired ACP channels to the Follower - and the Follower will not re-try to build these secure channels from its side unless the Decider shows up with a previously unknown GRASP announcement (e.g., on a different link or with a different address announced in GRASP).

6.13.5. ACP interfaces

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

6.13.5.1. ACP loopback interfaces

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

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

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

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

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

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

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

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

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

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

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

6.13.5.2. ACP virtual interfaces

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

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

6.13.5.2.1. ACP point-to-point virtual interfaces

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

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

6.13.5.2.2. ACP multi-access virtual interfaces

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

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

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

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

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

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

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

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

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

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

Assume a LAN with three ACP neighbors, Alice, Bob and Carol. Alice's ACP GRASP wants to send a link-local GRASP multicast message to Bob and Carol. If Alice's ACP emulates the LAN as per-peer, point-to-point virtual interfaces, one to Bob and one to Carol, Alice's ACP GRASP will send two copies of multicast GRASP messages: One to Bob and one to Carol. If Alice's ACP emulates a LAN via a multipoint virtual interface, Alice's ACP GRASP will send one packet to that interface and the ACP multipoint virtual interface will replicate the packet to each secure channel, one to Bob, one to Carol. The result is the same. The difference happens when Bob and Carol receive their packet. If they use ACP point-to-point virtual interfaces, their GRASP instance would forward the packet from Alice to each other as part of the GRASP flooding procedure. These packets are unnecessary and would be discarded by GRASP on receipt as duplicates (by use of the GRASP Session ID). If Bob and Carol's ACP would emulate a multi-access virtual interface, then this would not happen, because GRASP's flooding procedure does not replicate back packets to the interface that they were received from.

Note that link-local GRASP multicast messages are not sent directly as IPv6 link-local multicast UDP messages into ACP virtual interfaces, but instead into ACP GRASP virtual interfaces, that are layered on top of ACP virtual interfaces to add TCP reliability to link-local multicast GRASP messages. Nevertheless, these ACP GRASP virtual interfaces perform the same replication of message and, therefore, result in the same impact on flooding. See Section 6.9.2 for more details.

RPL does support operations and correct routing table construction across non-broadcast multi-access (NBMA) subnets. This is common when using many radio technologies. When such NBMA subnets are used, they MUST NOT be represented as ACP multi-access virtual interfaces because the replication of IPv6 link-local multicast messages will not reach all NBMA subnet neighbors. In result, GRASP message flooding would fail. Instead, each ACP secure channel across such an interface MUST be represented as a ACP point-to-point virtual interface. See also Appendix A.9.4.

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

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

7.1. Why (Benefits of ACP on L2 switches)

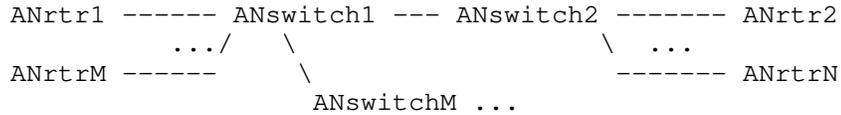


Figure 15: Topology with L2 ACP switches

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

If the discovery protocol used for the ACP is operating at the subnet level, every ACP router will see all other ACP routers on the LAN as neighbors and a full mesh of ACP channels will be built. If some or all of the AN switches are autonomic with the same discovery protocol, then the full mesh would include those switches as well.

A full mesh of ACP connections can create fundamental scale challenges. The number of security associations of the secure channel protocols will likely not scale arbitrarily, especially when they leverage platform accelerated encryption/decryption. Likewise, any other ACP operations (such as routing) needs to scale to the number of direct ACP neighbors. An ACP router with just 4 physical interfaces might be deployed into a LAN with hundreds of neighbors connected via switches. Introducing such a new unpredictable scaling factor requirement makes it harder to support the ACP on arbitrary platforms and in arbitrary deployments.

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

In the example above, consider ANswitch1 and ANswitchM are ACP capable, and ANswitch2 is not ACP capable. The desired ACP topology is that ANrtr1 and ANrtrM only have an ACP connection to ANswitch1, and that ANswitch1, ANrtr2, ANrtrN have a full mesh of ACP connection

amongst each other. ANswitch1 also has an ACP connection with ANswitchM and ANswitchM has ACP connections to anything else behind it.

7.2. How (per L2 port DULL GRASP)

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

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

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

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

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

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

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

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

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

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

8. Support for Non-ACP Components (Normative)

8.1. ACP Connect

8.1.1.1. Non-ACP Controller / NMS system

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

If the NMS host is not autonomic, i.e., it does not support autonomic negotiation of the ACP, then it can be brought into the ACP by explicit configuration. To support connections to adjacent non-ACP nodes, an ACP node SHOULD support "ACP connect" (sometimes also called "autonomic connect"):

"ACP connect" is an interface level configured workaround for connection of trusted non-ACP nodes to the ACP. The ACP node on which ACP connect is configured is called an "ACP edge node". With ACP connect, the ACP is accessible from those non-ACP nodes (such as NOC systems) on such an interface without those non-ACP nodes having to support any ACP discovery or ACP channel setup. This is also called "native" access to the ACP because to those NOC systems the interface looks like a normal network interface without any ACP secure channel that is encapsulating the traffic.

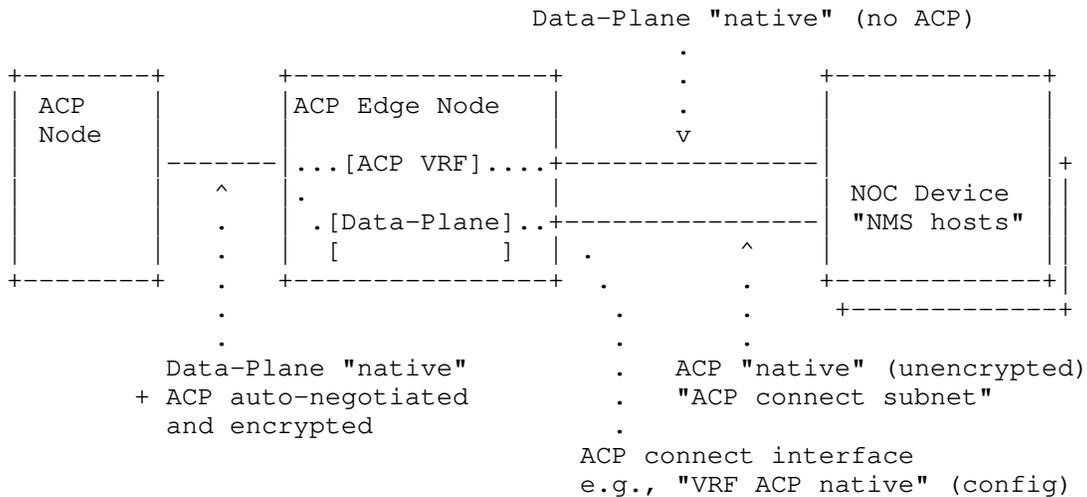


Figure 16: ACP connect

ACP connect has security consequences: All systems and processes connected via ACP connect have access to all ACP nodes on the entire ACP, without further authentication. Thus, the ACP connect interface and NOC systems connected to it needs to be physically controlled/secured. For this reason the mechanisms described here do explicitly not include options to allow for a non-ACP router to be connected across an ACP connect interface and addresses behind such a router routed inside the ACP.

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

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

An ACP connect interface SHOULD use an IPv6 address/prefix from the ACP Manual Addressing Sub-Scheme (Section 6.11.4), letting the operator configure for example only the Subnet-ID and having the node automatically assign the remaining part of the prefix/address. It SHOULD NOT use a prefix that is also routed outside the ACP so that the addresses clearly indicate whether it is used inside the ACP or not.

The prefix of ACP connect subnets MUST be distributed by the ACP edge node into the ACP routing protocol RPL. The NMS hosts MUST connect to prefixes in the ACP routing table via its ACP connect interface. In the simple case where the ACP uses only one ULA prefix and all ACP connect subnets have prefixes covered by that ULA prefix, NMS hosts can rely on [RFC6724] to determine longest match prefix routes towards its different interfaces, ACP and Data-Plane. With RFC6724, The NMS host will select the ACP connect interface for all addresses in the ACP because any ACP destination address is longest matched by the address on the ACP connect interface. If the NMS hosts ACP connect interface uses another prefix or if the ACP uses multiple ULA prefixes, then the NMS hosts require (static) routes towards the ACP interface for these prefixes.

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

To limit the security impact of ACP connect, nodes supporting it SHOULD implement a security mechanism to allow configuration/use of ACP connect interfaces only on nodes explicitly targeted to be deployed with it (those in physically secure locations such as a NOC). For example, the registrar could disable the ability to enable ACP connect on devices during enrollment and that property could only be changed through re-enrollment. See also Appendix A.9.5.

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

8.1.2. Software Components

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

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

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

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

8.1.3. Auto Configuration

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

The ACP edge node acts as the router towards the ACP on the ACP connect subnet, providing the (auto-)configured prefix for the ACP connect subnet and (auto-)configured routes into the ACP to NMS hosts and/or software components.

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

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

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

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

The ACP edge nodes that have a Vlong ACP address MAY allocate a subset of their /112 or /120 address prefix to ACP connect interface(s) to eliminate the need to non-autonomically configure/provision the address prefixes for such ACP connect interfaces.

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

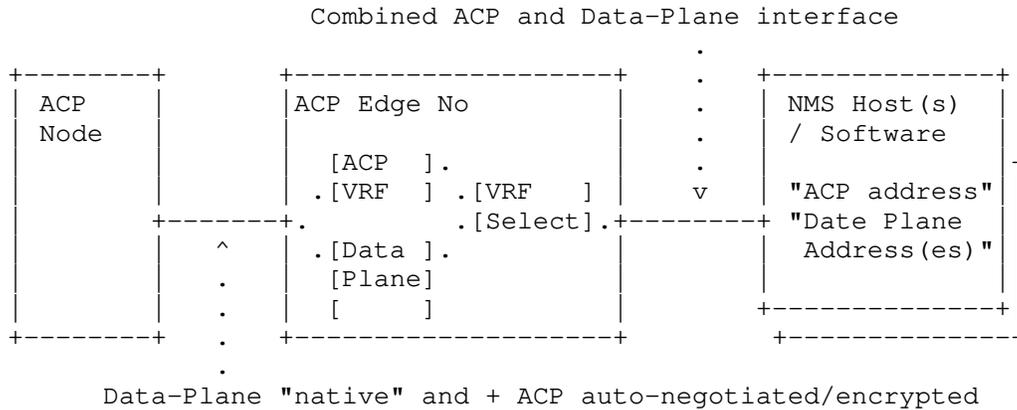


Figure 17: VRF select

Using two physical and/or virtual subnets (and therefore interfaces) into NMS Hosts (as per Section 8.1.1) or Software (as per Section 8.1.2) may be seen as additional complexity, for example with legacy NMS Hosts that support only one IP interface, or it may be insufficient to support [RFC4191] Type A or B host (see Section 8.1.3).

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

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

If the ACP carries multiple ULA prefixes or non-ULA ACP connect prefixes, then the correct source address selection becomes even more problematic.

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

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

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

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

8.1.5. Use of GRASP

GRASP can and should be possible to use across ACP connect interfaces, especially in the architectural correct solution when it is used as a mechanism to connect Software (e.g., ASA or legacy NMS applications) to the ACP.

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

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

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

8.2. Connecting ACP islands over Non-ACP L3 networks (Remote ACP neighbors)

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

8.2.1. Configured Remote ACP neighbor

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

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

Figure 18: Parameters for remote ACP neighbors

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

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

- * The local and remote address can be IPv4 or IPv6 and are typically global scope addresses.
- * The VRF across which the connection is built (and in which local-addr exists) can to be specified. If vrf is not specified, it is the default VRF on the node. In DULL GRASP the VRF is implied by the interface across which DULL GRASP operates.
- * If local address is "any", the local address used when initiating a secure channel connection is decided by source address selection ([RFC6724] for IPv6). As a responder, the connection listens on all addresses of the node in the selected VRF.
- * Configuration of port is only required for methods where no defaults exist (e.g., "DTLS").

- * If remote address is "any", the connection is only a responder. It is a "hub" that can be used by multiple remote peers to connect simultaneously - without having to know or configure their addresses. Example: Hub site for remote "spoke" sites reachable over the Internet.
- * Pmtu should be configurable to overcome issues/limitations of Path MTU Discovery (PMTUD).
- * IKEv2/IPsec to remote peers should support the optional NAT Traversal (NAT-T) procedures.

8.2.2. Tunneled Remote ACP Neighbor

An IPinIP, GRE or other form of pre-existing tunnel is configured between two remote ACP peers and the virtual interfaces representing the tunnel are configured for "ACP enable". This will enable IPv6 link local addresses and DULL on this tunnel. In result, the tunnel is used for normal "L2 adjacent" candidate ACP neighbor discovery with DULL and secure channel setup procedures described in this document.

Tunneled Remote ACP Neighbor requires two encapsulations: the configured tunnel and the secure channel inside of that tunnel. This makes it in general less desirable than Configured Remote ACP Neighbor. Benefits of tunnels are that it may be easier to implement because there is no change to the ACP functionality - just running it over a virtual (tunnel) interface instead of only native interfaces. The tunnel itself may also provide PMTUD while the secure channel method may not. Or the tunnel mechanism is permitted/possible through some firewall while the secure channel method may not.

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

8.2.3. Summary

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

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

9. ACP Operations (Informative)

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

- * Section 9.1 describes recommended operator diagnostics capabilities of ACP nodes.
- * Section 9.2 describes high level how an ACP registrar needs to work, what its configuration parameters are and specific issues impacting the choices of deployment design due to renewal and revocation issues. It describes a model where ACP Registrars have their own sub-CA to provide the most distributed deployment option for ACP Registrars, and it describes considerations for centralized policy control of ACP Registrar operations.
- * Section 9.3 describes suggested ACP node behavior and operational interfaces (configuration options) to manage the ACP in so-called greenfield devices (previously unconfigured) and brownfield devices (preconfigured).

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

9.1. ACP (and BRSKI) Diagnostics

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

Basic standardized diagnostics would require support for (yang) models representing the complete (auto-)configuration and operational state of all components: GRASP, ACP and the infrastructure used by them: TLS/DTLS, IPsec, certificates, TA, time, VRF and so on. While necessary, this is not sufficient:

Simply representing the state of components does not allow operators to quickly take action - unless they do understand how to interpret the data, and that can mean a requirement for deep understanding of all components and how they interact in the ACP/ANI.

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

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

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

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

Q: Is ACP operational to accept neighbor connections:

- * Check if any potentially necessary configuration to make ACP/ANI operational are correct (see Section 9.3 for a discussion of such commands).
- * Does the system time look reasonable, or could it be the default system time after clock chip battery failure (certificate checks depend on reasonable notion of time)?.

- * Does the node have keying material - domain certificate, TA certificates, ...>
- * If no keying material and ANI is supported/enabled, check the state of BRSKI (not detailed in this example).
- * Check the validity of the domain certificate:
 - Does the certificate validate against the TA?
 - Has it been revoked?
 - Was the last scheduled attempt to retrieve a CRL successful (e.g., do we know that our CRL information is up to date).
 - Is the certificate valid: validity start time in the past, expiration time in the future?
 - Does the certificate have a correctly formatted acp-node-name field?
- * Was the ACP VRF successfully created?
- * Is ACP enabled on one or more interfaces that are up and running?

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

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

- * Is the interface physically up? Does it have an IPv6 link-local address?
- * Is it enabled for ACP?
- * Do we successfully send DULL GRASP messages to the interface (link layer errors)?
- * Do we receive DULL GRASP messages on the interface? If not, some intervening L2 equipment performing bad MLD snooping could have caused problems. Provide e.g., diagnostics of the MLD querier IPv6 and MAC address.
- * Do we see the ACP objective in any DULL GRASP message from that interface? Diagnose the supported secure channel methods.
- * Do we know the MAC address of the neighbor with the ACP objective? If not, diagnose SLAAC/ND state.
- * When did we last attempt to build an ACP secure channel to the neighbor?
- * If it failed, why:
 - Did the neighbor close the connection on us or did we close the connection on it because the domain certificate membership failed?
 - If the neighbor closed the connection on us, provide any error diagnostics from the secure channel protocol.
 - If we failed the attempt, display our local reason:
 - o There was no common secure channel protocol supported by the two neighbors (this could not happen on nodes supporting this specification because it mandates common support for IPsec).

- o The ACP certificate membership check (Section 6.2.3) fails:
 - + The neighbor's certificate is not signed directly or indirectly by one of the nodes TA. Provide diagnostics which TA it has (can identify whom the device belongs to).
 - + The neighbor's certificate does not have the same domain (or no domain at all). Diagnose domain-name and potentially other cert info.
 - + The neighbor's certificate has been revoked or could not be authenticated by OCSP.
 - + The neighbor's certificate has expired - or is not yet valid.
- Any other connection issues in e.g., IKEv2 / IPsec, DTLS?.

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

- * Are there one or more active ACP neighbors with secure channels?
- * Is the RPL routing protocol for the ACP running?
- * Is there a default route to the root in the ACP routing table?
- * Is there for each direct ACP neighbor not reachable over the ACP virtual interface to the root a route in the ACP routing table?
- * Is ACP GRASP running?
- * Is at least one SRV.est objective cached (to support certificate renewal)?
- * Is there at least one BRSKI registrar objective cached (in case BRSKI is supported)
- * Is BRSKI proxy operating normally on all interfaces where ACP is operating?
- * ...

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

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

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

1. BRSKI does not allow for a neighboring device to identify the pledges IDevID certificate. Only the selected BRSKI registrar can do this, but it may be difficult to disseminate information about undesired pledges from those BRSKI registrars to locations/nodes where information about those pledges is desired.
2. LLDP disseminates information about nodes to their immediate neighbors, such as node model/type/software and interface name/number of the connection. This information is often helpful or even necessary in network diagnostics. It can equally be considered to be too insecure to make this information available unprotected to all possible neighbors.

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

9.1.1. Secure Channel Peer diagnostics

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

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

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

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

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

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

9.2. ACP Registrars

As described in Section 6.11.7, the ACP addressing mechanism is designed to enable lightweight, distributed and uncoordinated ACP registrars that are providing ACP address prefixes to candidate ACP nodes by enrolling them with an ACP certificate into an ACP domain via any appropriate mechanism/protocol, automated or not.

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

9.2.1. Registrar interactions

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

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

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

To reach a TA that has no ACP connectivity, the ACP registrar would use the Data-Plane. ACP and Data-Plane in an ACP registrar could (and by default should be) completely isolated from each other at the network level. Only applications such as the ACP registrar would need the ability for their transport stacks to access both.

In non-autonomic enrollment options, the Data-Plane between a ACP registrar and the candidate ACP node needs to be configured first. This includes the ACP registrar and the candidate ACP node. Then any appropriate set of protocols can be used between ACP registrar and candidate ACP node to discover the other side, and then connect and enroll (configure) the candidate ACP node with an ACP certificate. NETCONF ZeroTouch ([RFC8572]) is an example protocol that could be used for this. BRSKI using optional discovery mechanisms is equally a possibility for candidate ACP nodes attempting to be enrolled across non-ACP networks, such as the Internet.

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

An ACP registrar could operate EST for ACP certificate renewal and/or act as a CRL Distribution point. A node performing these services does not need to support performing (initial) enrollment, but it does require the same above described connectivity as an ACP registrar: via the ACP to ACP nodes and via the Data-Plane to the TA and other sources of CRL information.

9.2.2. Registrar Parameter

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

- * A URL to the TA and credentials so that the ACP registrar can let the TA sign candidate ACP node certificates.
- * The ACP domain-name.
- * The Registrar-ID to use. This could default to a MAC address of the ACP registrar.

- * For recovery, the next-useable Node-IDs for zone (Zone-ID=0) sub-addressing scheme, for Vlong /112 and for Vlong /120 sub-addressing scheme. These IDs would only need to be provisioned after recovering from a crash. Some other mechanism would be required to remember these IDs in a backup location or to recover them from the set of currently known ACP nodes.
- * Policies if candidate ACP nodes should receive a domain certificate or not, for example based on the devices IDevID certificate as in BRSKI. The ACP registrar may have a whitelist or blacklist of devices [X.520] "serialNumbers" attribute in the subject field distinguished name encoding from their IDevID certificate.
- * Policies what type of address prefix to assign to a candidate ACP devices, based on likely the same information.
- * For BRSKI or other mechanisms using vouchers: Parameters to determine how to retrieve vouchers for specific type of secure bootstrap candidate ACP nodes (such as MASA URLs), unless this information is automatically learned such as from the IDevID certificate of candidate ACP nodes (as defined in BRSKI).

9.2.3. Certificate renewal and limitations

When an ACP node renews/rekeys its certificate, it may end up doing so via a different registrar (e.g., EST server) than the one it originally received its ACP certificate from, for example because that original ACP registrar is gone. The ACP registrar through which the renewal/rekeying is performed would by default trust the acp-node-name from the ACP nodes current ACP certificate and maintain this information so that the ACP node maintains its ACP address prefix. In EST renewal/rekeying, the ACP nodes current ACP certificate is signaled during the TLS handshake.

This simple scenario has two limitations:

1. The ACP registrars cannot directly assign certificates to nodes and therefore needs an "online" connection to the TA.
2. Recovery from a compromised ACP registrar is difficult. When an ACP registrar is compromised, it can insert for example a conflicting acp-node-name and create thereby an attack against other ACP nodes through the ACP routing protocol.

Even when such a malicious ACP registrar is detected, resolving the problem may be difficult because it would require identifying all the wrong ACP certificates assigned via the ACP registrar after it was compromised. And without additional centralized tracking of assigned certificates there is no way to do this.

9.2.4. ACP Registrars with sub-CA

In situations, where either of the above two limitations are an issue, ACP registrars could also be sub-CAs. This removes the need for connectivity to a TA whenever an ACP node is enrolled, and reduces the need for connectivity of such an ACP registrar to a TA to only those times when it needs to renew its own certificate. The ACP registrar would also now use its own (sub-CA) certificate to enroll and sign the ACP nodes certificates, and therefore it is only necessary to revoke a compromised ACP registrars sub-CA certificate. Alternatively one can let it expire and not renew it, when the certificate of the sub-CA is appropriately short-lived.

As the ACP domain membership check verifies a peer ACP node's ACP certificate trust chain, it will also verify the signing certificate which is the compromised/revoked sub-CA certificate. Therefore, ACP domain membership for an ACP node enrolled from a compromised and discovered ACP registrar will fail.

ACP nodes enrolled by a compromised ACP registrar would automatically fail to establish ACP channels and ACP domain certificate renewal via EST and therefore revert to their role as a candidate ACP members and attempt to get a new ACP certificate from an ACP registrar - for example, via BRSKI. In result, ACP registrars that have an associated sub-CA makes isolating and resolving issues with compromised registrars easier.

Note that ACP registrars with sub-CA functionality also can control the lifetime of ACP certificates easier and therefore also be used as a tool to introduce short lived certificates and not rely on CRL, whereas the certificates for the sub-CAs themselves could be longer lived and subject to CRL.

9.2.5. Centralized Policy Control

When using multiple, uncoordinated ACP registrars, several advanced operations are potentially more complex than with a single, resilient policy control backend, for example including but not limited to:

- * Which candidate ACP node is permitted or not permitted into an ACP domain. This may not be a decision to be taken upfront, so that a policy per "serialNumber" attribute in the subject field distinguished name encoding can be loaded into every ACP registrar. Instead, it may better be decided in real-time including potentially a human decision in a NOC.
- * Tracking of all enrolled ACP nodes and their certificate information. For example, in support of revoking individual ACP nodes certificates.

- * More flexible policies what type of address prefix or even what specific address prefix to assign to a candidate ACP node.

These and other operations could be introduced more easily by introducing a centralized Policy Management System (PMS) and modifying ACP registrar behavior so that it queries the PMS for any policy decision occurring during the candidate ACP node enrollment process and/or the ACP node certificate renewal process. For example, which ACP address prefix to assign. Likewise the ACP registrar would report any relevant state change information to the PMS as well, for example when a certificate was successfully enrolled onto a candidate ACP node.

9.3. Enabling and disabling ACP/ANI

Both ACP and BRSKI require interfaces to be operational enough to support sending/receiving their packets. In node types where interfaces are by default (e.g., without operator configuration) enabled, such as most L2 switches, this would be less of a change in behavior than in most L3 devices (e.g. routers), where interfaces are by default disabled. In almost all network devices it is common though for configuration to change interfaces to a physically disabled state and that would break the ACP.

In this section, we discuss a suggested operational model to enable/disable interfaces and nodes for ACP/ANI in a way that minimizes the risk of operator action to break the ACP in this way, and that also minimizes operator surprise when ACP/ANI becomes supported in node software.

9.3.1. Filtering for non-ACP/ANI packets

Whenever this document refers to enabling an interface for ACP (or BRSKI), it only requires to permit the interface to send/receive packets necessary to operate ACP (or BRSKI) - but not any other Data-Plane packets. Unless the Data-Plane is explicitly configured/enabled, all packets not required for ACP/BRSKI should be filtered on input and output:

Both BRSKI and ACP require link-local only IPv6 operations on interfaces and DULL GRASP. IPv6 link-local operations means the minimum signaling to auto-assign an IPv6 link-local address and talk to neighbors via their link-local address: SLAAC (Stateless Address Auto-Configuration - [RFC4862]) and ND (Neighbor Discovery - [RFC4861]). When the device is a BRSKI pledge, it may also require TCP/TLS connections to BRSKI proxies on the interface. When the device has keying material, and the ACP is running, it requires DULL GRASP packets and packets necessary for the secure-channel mechanism

it supports, e.g., IKEv2 and IPsec ESP packets or DTLS packets to the IPv6 link-local address of an ACP neighbor on the interface. It also requires TCP/TLS packets for its BRSKI proxy functionality, if it does support BRSKI.

9.3.2. Admin Down State

Interfaces on most network equipment have at least two states: "up" and "down". These may have product specific names. "down" for example could be called "shutdown" and "up" could be called "no shutdown". The "down" state disables all interface operations down to the physical level. The "up" state enables the interface enough for all possible L2/L3 services to operate on top of it and it may also auto-enable some subset of them. More commonly, the operations of various L2/L3 services is controlled via additional node-wide or interface level options, but they all become only active when the interface is not "down". Therefore, an easy way to ensure that all L2/L3 operations on an interface are inactive is to put the interface into "down" state. The fact that this also physically shuts down the interface is in many cases just a side effect, but it may be important in other cases (see below, Section 9.3.2.2).

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

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

To provide ACP/ANI resilience against such operator misconfiguration, this document recommends to separate the "down" state of interfaces into an "admin down" state where the physical layer is kept running and ACP/ANI can use the interface and a "physical down" state. Any existing "down" configurations would map to "admin down". In "admin down", any existing L2/L3 services of the Data-Plane should see no difference to "physical down" state. To ensure that no Data-Plane packets could be sent/received, packet filtering could be established automatically as described above in Section 9.3.1.

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

As necessary (see discussion below) new configuration options could be introduced to issue "physical down". The options should be provided with additional checks to minimize the risk of issuing them in a way that breaks the ACP without automatic restoration. For example, they could be denied to be issued from a control connection (NETCONF/SSH) that goes across the interface itself ("do not disconnect yourself"). Or they could be performed only temporary and only be made permanent with additional later reconfirmation.

In the following sub-sections important aspects to the introduction of "admin down" state are discussed.

9.3.2.1. Security

Interfaces are physically brought down (or left in default down state) as a form of security. "Admin down" state as described above provides also a high level of security because it only permits ACP/ANI operations which are both well secured. Ultimately, it is subject to security review for the deployment whether "admin down" is a feasible replacement for "physical down".

The need to trust the security of ACP/ANI operations needs to be weighed against the operational benefits of permitting this: Consider the typical example of a CPE (customer premises equipment) with no on-site network expert. User ports are in physical down state unless explicitly configured not to be. In a misconfiguration situation, the uplink connection is incorrectly plugged into such as user port. The device is disconnected from the network and therefore no diagnostics from the network side is possible anymore. Alternatively, all ports default to "admin down". The ACP (but not the Data-Plane) would still automatically form. Diagnostics from the network side is possible and operator reaction could include to either make this port the operational uplink port or to instruct re-cabling. Security wise, only ACP/ANI could be attacked, all other functions are filtered on interfaces in "admin down" state.

9.3.2.2. Fast state propagation and Diagnostics

"Physical down" state propagates on many interface types (e.g., Ethernet) to the other side. This can trigger fast L2/L3 protocol reaction on the other side and "admin down" would not have the same (fast) result.

Bringing interfaces to "physical down" state is to the best of our knowledge always a result of operator action, but today, never the result of autonomic L2/L3 services running on the nodes. Therefore, one option is to change the operator action to not rely on link-state propagation anymore. This may not be possible when both sides are under different operator control, but in that case it is unlikely that the ACP is running across the link and actually putting the interface into "physical down" state may still be a good option.

Ideally, fast physical state propagation is replaced by fast software driven state propagation. For example, a DULL GRASP "admin-state" objective could be used to auto configure a Bidirectional Forwarding Protocol (BFD, [RFC5880]) session between the two sides of the link that would be used to propagate the "up" vs. admin down state.

Triggering physical down state may also be used as a mean of diagnosing cabling in the absence of easier methods. It is more complex than automated neighbor diagnostics because it requires coordinated remote access to both (likely) sides of a link to determine whether up/down toggling will cause the same reaction on the remote side.

See Section 9.1 for a discussion about how LLDP and/or diagnostics via GRASP could be used to provide neighbor diagnostics, and therefore hopefully eliminating the need for "physical down" for neighbor diagnostics - as long as both neighbors support ACP/ANI.

9.3.2.3. Low Level Link Diagnostics

"Physical down" is performed to diagnose low-level interface behavior when higher layer services (e.g., IPv6) are not working. Especially Ethernet links are subject to a wide variety of possible wrong configuration/cablings if they do not support automatic selection of variable parameters such as speed (10/100/1000 Mbps), crossover (Auto-MDIX) and connector (fiber, copper - when interfaces have multiple but can only enable one at a time). The need for low level link diagnostic can therefore be minimized by using fully auto configuring links.

In addition to "Physical down", low level diagnostics of Ethernet or other interfaces also involve the creation of other states on interfaces, such as physical Loopback (internal and/or external) or bringing down all packet transmissions for reflection/cable-length measurements. Any of these options would disrupt ACP as well.

In cases where such low-level diagnostics of an operational link is desired but where the link could be a single point of failure for the ACP, ASA on both nodes of the link could perform a negotiated

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

9.3.2.4. Power Consumption Issues

Power consumption of "physical down" interfaces, may be significantly lower than those in "admin down" state, for example on long-range fiber interfaces. Bringing up interfaces, for example to probe reachability, may also consume additional power. This can make these type of interfaces inappropriate to operate purely for the ACP when they are not currently needed for the Data-Plane.

9.3.3. Interface level ACP/ANI enable

The interface level configuration option "ACP enable" enables ACP operations on an interface, starting with ACP neighbor discovery via DULL GRAP. The interface level configuration option "ANI enable" on nodes supporting BRSKI and ACP starts with BRSKI pledge operations when there is no domain certificate on the node. On ACP/BRSKI nodes, "ACP enable" may not need to be supported, but only "ANI enable". Unless overridden by global configuration options (see later), "ACP/ANI enable" will result in "down" state on an interface to behave as "admin down".

9.3.4. Which interfaces to auto-enable?

(Section 6.4) requires that "ACP enable" is automatically set on native interfaces, but not on non-native interfaces (reminder: a native interface is one that exists without operator configuration action such as physical interfaces in physical devices).

Ideally, ACP enable is set automatically on all interfaces that provide access to additional connectivity that allows to reach more nodes of the ACP domain. The best set of interfaces necessary to achieve this is not possible to determine automatically. Native interfaces are the best automatic approximation.

Consider an ACP domain of ACP nodes transitively connected via native interfaces. A Data-Plane tunnel between two of these nodes that are non-adjacent is created and "ACP enable" is set for that tunnel. ACP RPL sees this tunnel as just as a single hop. Routes in the ACP would use this hop as an attractive path element to connect regions adjacent to the tunnel nodes. In result, the actual hop-by-hop paths used by traffic in the ACP can become worse. In addition, correct forwarding in the ACP now depends on correct Data-Plane forwarding config including QoS, filtering and other security on the Data-Plane path across which this tunnel runs. This is the main issue why "ACP/ANI enable" should not be set automatically on non-native interfaces.

If the tunnel would connect two previously disjoint ACP regions, then it likely would be useful for the ACP. A Data-Plane tunnel could also run across nodes without ACP and provide additional connectivity for an already connected ACP network. The benefit of this additional ACP redundancy has to be weighed against the problems of relying on the Data-Plane. If a tunnel connects two separate ACP regions: how many tunnels should be created to connect these ACP regions reliably enough? Between which nodes? These are all standard tunneled network design questions not specific to the ACP, and there are no generic fully automated answers.

Instead of automatically setting "ACP enable" on these type of interfaces, the decision needs to be based on the use purpose of the non-native interface and "ACP enable" needs to be set in conjunction with the mechanism through which the non-native interface is created/configured.

In addition to explicit setting of "ACP/ANI enable", non-native interfaces also need to support configuration of the ACP RPL cost of the link - to avoid the problems of attracting too much traffic to the link as described above.

Even native interfaces may not be able to automatically perform BRSKI or ACP because they may require additional operator input to become operational. Example include DSL interfaces requiring PPPoE credentials or mobile interfaces requiring credentials from a SIM card. Whatever mechanism is used to provide the necessary config to the device to enable the interface can also be expanded to decide on whether or not to set "ACP/ANI enable".

The goal of automatically setting "ACP/ANI enable" on interfaces (native or not) is to eliminate unnecessary "touches" to the node to make its operation as much as possible "zero-touch" with respect to ACP/ANI. If there are "unavoidable touches" such a creating/configuring a non-native interface or provisioning credentials for a native interface, then "ACP/ANI enable" should be added as an option

to that "touch". If a wrong "touch" is easily fixed (not creating another high-cost touch), then the default should be not to enable ANI/ACP, and if it is potentially expensive or slow to fix (e.g., parameters on SIM card shipped to remote location), then the default should be to enable ACP/ANI.

9.3.5. Node Level ACP/ANI enable

A node level command "ACP/ANI enable [up-if-only]" enables ACP or ANI on the node (ANI = ACP + BRSKI). Without this command set, any interface level "ACP/ANI enable" is ignored. Once set, ACP/ANI will operate an interface where "ACP/ANI enable" is set. Setting of interface level "ACP/ANI enable" is either automatic (default) or explicit through operator action as described in the previous section.

If the option "up-if-only" is selected, the behavior of "down" interfaces is unchanged, and ACP/ANI will only operate on interfaces where "ACP/ANI enable" is set and that are "up". When it is not set, then "down" state of interfaces with "ACP/ANI enable" is modified to behave as "admin down".

9.3.5.1. Brownfield nodes

A "brownfield" node is one that already has a configured Data-Plane.

Executing global "ACP/ANI enable [up-if-only]" on each node is the only command necessary to create an ACP across a network of brownfield nodes once all the nodes have a domain certificate. When BRSKI is used ("ANI enable"), provisioning of the certificates only requires set-up of a single BRSKI registrar node which could also implement a CA for the network. This is the simplest way to introduce ACP/ANI into existing (== brownfield) networks.

The need to explicitly enable ACP/ANI is especially important in brownfield nodes because otherwise software updates may introduce support for ACP/ANI: Automatic enablement of ACP/ANI in networks where the operator does not only not want ACP/ANI but where the operator likely never even heard of it could be quite irritating to the operator. Especially when "down" behavior is changed to "admin down".

Automatically setting "ANI enable" on brownfield nodes where the operator is unaware of BRSKI and MASA operations could also be an unlikely but then critical security issue. If an attacker could impersonate the operator and register as the operator at the MASA or otherwise get hold of vouchers and can get enough physical access to the network so pledges would register to an attacking registrar, then the attacker could gain access to the ACP, and through the ACP gain access to the Data-Plane.

In networks where the operator explicitly wants to enable the ANI this could not happen, because the operator would create a BRSKI registrar that would discover attack attempts, and the operator would be setting up his registrar with the MASA. Nodes requiring "ownership vouchers" would not be subject to that attack. See [I-D.ietf-anima-bootstrapping-keyinfra] for more details. Note that a global "ACP enable" alone is not subject to these type of attacks, because it always depends on some other mechanism first to provision domain certificates into the device.

9.3.5.2. Greenfield nodes

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

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

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

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

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

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

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

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

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

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

9.3.6. Undoing ANI/ACP enable

Disabling ANI/ACP by undoing "ACP/ANI enable" is a risk for the reliable operations of the ACP if it can be executed by mistake or unauthorized. This behavior could be influenced through some additional (future) property in the certificate (e.g., in the acp-node-name extension field): In an ANI deployment intended for convenience, disabling it could be allowed without further constraints. In an ANI deployment considered to be critical more checks would be required. One very controlled option would be to not permit these commands unless the domain certificate has been revoked or is denied renewal. Configuring this option would be a parameter on the BRSKI registrar(s). As long as the node did not receive a domain certificate, undoing "ANI/ACP enable" should not have any additional constraints.

9.3.7. Summary

Node-wide "ACP/ANI enable [up-if-only]" commands enable the operation of ACP/ANI. This is only auto-enabled on ANI greenfield devices, otherwise it must be configured explicitly.

If the option "up-if-only" is not selected, interfaces enabled for ACP/ANI interpret "down" state as "admin down" and not "physical down". In "admin-down" all non-ACP/ANI packets are filtered, but the physical layer is kept running to permit ACP/ANI to operate.

(New) commands that result in physical interruption ("physical down", "loopback") of ACP/ANI enabled interfaces should be built to protect continuance or reestablishment of ACP as much as possible.

Interface level "ACP/ANI enable" control per-interface operations. It is enabled by default on native interfaces and has to be configured explicitly on other interfaces.

Disabling "ACP/ANI enable" global and per-interface should have additional checks to minimize undesired breakage of ACP. The degree of control could be a domain wide parameter in the domain certificates.

9.4. Partial or Incremental adoption

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

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

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

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

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

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

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

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

9.5. Configuration and the ACP (summary)

There is no desirable configuration for the ACP. Instead, all parameters that need to be configured in support of the ACP are limitations of the solution, but they are only needed in cases where not all components are made autonomic. Wherever this is necessary, it relies on pre-existing mechanisms for configuration such as CLI or YANG ([RFC7950]) data models.

The most important examples of such configuration include:

- * When ACP nodes do not support an autonomic way to receive an ACP certificate, for example BRSKI, then such certificate needs to be configured via some pre-existing mechanisms outside the scope of this specification. Today, router have typically a variety of mechanisms to do this.
- * Certificate maintenance requires PKI functions. Discovery of these functions across the ACP is automated (see Section 6.2.5), but their configuration is not.

- * When non-ACP capable nodes such as pre-existing NMS need to be physically connected to the ACP, the ACP node to which they attach needs to be configured with ACP-connect according to Section 8.1. It is also possible to use that single physical connection to connect both to ACP and the Data-Plane of the network as explained in Section 8.1.4.
- * When devices are not autonomically bootstrapped, explicit configuration to enable the ACP needs to be applied. See Section 9.3.
- * When the ACP needs to be extended across interfaces other than L2, the ACP as defined in this document cannot autodiscover candidate neighbors automatically. Remote neighbors need to be configured, see Section 8.2.

Once the ACP is operating, any further configuration for the Data-Plane can be configured more reliably across the ACP itself because the ACP provides addressing and connectivity (routing) independent of the Data-Plane itself. For this, the configuration methods simply need to also allow to operate across the ACP VRF - NETCONF, SSH or any other method.

The ACP also provides additional security through its hop-by-hop encryption for any such configuration operations: Some legacy configuration methods (SNMP, TFTP, HTTP) may not use end-to-end encryption, and most of the end-to-end secured configuration methods still allow for easy passive observation along the path about configuration taking place (transport flows, port numbers, IP addresses).

The ACP can and should equally be used as the transport to configure any of the aforementioned non-autonomic components of the ACP, but in that case, the same caution needs to be exercised as with Data-Plane configuration without ACP: Misconfiguration may cause the configuring entity to be disconnected from the node it configures - for example when incorrectly unconfiguring a remote ACP neighbor through which the configured ACP node is reached.

10. Summary: Benefits (Informative)

10.1. Self-Healing Properties

The ACP is self-healing:

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

- * When any changes happen in the topology, the routing protocol used in the ACP will automatically adapt to the changes and will continue to provide reachability to all nodes.
- * The ACP tracks the validity of peer certificates and tears down ACP secure channels when a peer certificate has expired. When short-lived certificates with lifetimes in the order of OCSP/CRL refresh times are used, then this allows for removal of invalid peers (whose certificate was not renewed) at similar speeds as when using OCSP/CRL. The same benefit can be achieved when using CRL/OCSP, periodically refreshing the revocation information and also tearing down ACP secure channels when the peer's (long-lived) certificate is revoked. There is no requirement against ACP implementations to require this enhancement though to keep the mandatory implementations simpler.

The ACP can also sustain network partitions and mergers. Practically all ACP operations are link local, where a network partition has no impact. Nodes authenticate each other using the domain certificates to establish the ACP locally. Addressing inside the ACP remains unchanged, and the routing protocol inside both parts of the ACP will lead to two working (although partitioned) ACPs.

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

Highly resilient ACP designs can be built by using ACP Registrars with embedded sub-CA, as outlined in Section 9.2.4. As long as a partition is left with one or more of such ACP Registrars, it can continue to enroll new candidate ACP nodes as long as the ACP Registrar's sub-CA certificate does not expire. Because the ACP addressing relies on unique Registrar-IDs, a later re-merge of partitions will also not cause problems with ACP addresses assigned during partitioning.

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

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

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

10.2. Self-Protection Properties

10.2.1. From the outside

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

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

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

The ACP also serves as protection (through authentication and encryption) for protocols relevant to OAM that may not have secured protocol stack options or where implementation or deployment of those options fail on some vendor/product/customer limitations. This includes protocols such as SNMP ([RFC3411]), NTP ([RFC5905]), PTP ([IEEE-1588-2008]), DNS ([RFC3596]), DHCPv6 ([RFC3315]), syslog ([RFC3164]), RADIUS ([RFC2865]), Diameter ([RFC6733]), TACACS ([RFC1492]), IPFIX ([RFC7011]), Netflow ([RFC3954]) – just to name a few. Not all of these protocol references are necessarily the latest version of protocols but versions that are still widely deployed.

Protection via the ACP secure hop-by-hop channels for these protocols is meant to be only a stopgap though: The ultimate goal is for these and other protocols to use end-to-end encryption utilizing the domain certificate and rely on the ACP secure channels primarily for zero-touch reliable connectivity, but not primarily for security.

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

10.2.2. From the inside

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

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

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

See Appendix A.9.8 for further considerations how to avoid and deal with compromised nodes.

10.3. The Administrator View

An ACP is self-forming, self-managing and self-protecting, therefore has minimal dependencies on the administrator of the network. Specifically, since it is (intended to be) independent of configuration, there is only limited scope for configuration errors on the ACP itself. The administrator may have the option to enable or disable the entire approach, but detailed configuration is not possible. This means that the ACP must not be reflected in the running configuration of nodes, except a possible on/off switch (and even that is undesirable).

While configuration (except for Section 8 and Section 9.2) is not possible, an administrator must have full visibility of the ACP and all its parameters, to be able to do trouble-shooting. Therefore, an ACP must support all show and debug options, as for any other network function. Specifically, a network management system or controller must be able to discover the ACP, and monitor its health. This visibility of ACP operations must clearly be separated from visibility of Data-Plane so automated systems will never have to deal with ACP aspects unless they explicitly desire to do so.

Since an ACP is self-protecting, a node not supporting the ACP, or without a valid domain certificate cannot connect to it. This means that by default a traditional controller or network management system cannot connect to an ACP. See Section 8.1.1 for more details on how to connect an NMS host into the ACP.

11. Security Considerations

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

The self-protecting property does not include workarounds for non-autonomic components as explained in Section 8. See Section 10.2 for details of how the ACP protects itself against attacks from the outside and to a more limited degree from the inside as well.

However, the security of the ACP depends on a number of other factors:

- * The usage of domain certificates depends on a valid supporting PKI infrastructure. If the chain of trust of this PKI infrastructure is compromised, the security of the ACP is also compromised. This is typically under the control of the network administrator.
- * ACP nodes receive their certificates from ACP registrars. These ACP registrars are security critical dependencies of the ACP: Procedures and protocols for ACP registrars are outside the scope of this specification as explained in Section 6.11.7.1, only requirements against the resulting ACP certificates are specified.
- * Every ACP registrar (for enrollment of ACP certificates) and ACP EST server (for renewal of ACP certificates) is a security critical entity and its protocols are security critical protocols. Both need to be hardened against attacks, similar to a CA and its protocols. A malicious registrar can enroll malicious nodes to an ACP network (if the CA delegates this policy to the registrar) or break ACP routing for example by assigning duplicate ACP address assignment to ACP nodes via their ACP certificates.
- * ACP nodes that are ANI nodes rely on BRSKI as the protocol for ACP registrars. For ANI type ACP nodes, the security considerations of BRSKI apply. It enables automated, secure enrollment of ACP certificates.
- * BRSKI and potentially other ACP registrar protocol options require that nodes have an (X.509v3 based) IDevID. IDevIDs are an option for ACP registrars to securely identify candidate ACP nodes that should be enrolled into an ACP domain.

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

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

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

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

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

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

There is no prevention of source-address spoofing inside the ACP. This implies that if an attacker gains access to the ACP, it can spoof all addresses inside the ACP and fake messages from any other node. New protocol/services run across the ACP should therefore use end-to-end authentication inside the ACP. This is already done by GRASP as specified in this document.

The ACP is designed to enable automation of current network management and future autonomic peer-to-peer/distributed network automation. Any ACP member can send ACP IPv6 packet to other ACP members and announce via ACP GRASP services to all ACP members without dependency against centralized components.

The ACP relies on peer-to-peer authentication and authorization using ACP certificates. This security model is necessary to enable the autonomic ad-hoc any-to-any connectivity between ACP nodes. It provides infrastructure protection through hop by hop authentication and encryption - without relying on third parties. For any services

where this complete autonomic peer-to-peer group security model is appropriate, the ACP certificate can also be used unchanged. For example, for any type of Data-Plane routing protocol security.

This ACP security model is designed primarily to protect against attack from the outside, but not against attacks from the inside. To protect against spoofing attacks from compromised on-path ACP nodes, end-to-end encryption inside the ACP is used by new ACP signaling: GRASP across the ACP using TLS. The same is expected from any non-legacy services/protocols using the ACP. Because no group-keys are used, there is no risk for impacted nodes to access end-to-end encrypted traffic from other ACP nodes.

Attacks from impacted ACP nodes against the ACP are more difficult than against the Data-Plane because of the autoconfiguration of the ACP and the absence of configuration options that could be abused that allow to change/break ACP behavior. This is excluding configuration for workaround in support of non-autonomic components.

Mitigation against compromised ACP members is possible through standard automated certificate management mechanisms including revocation and non-renewal of short-lived certificates. In this version of the specification, there are no further optimization of these mechanisms defined for the ACP (but see Appendix A.9.8).

Higher layer service built using ACP certificates should not solely rely on undifferentiated group security when another model is more appropriate/more secure. For example, central network configuration relies on a security model where only few especially trusted nodes are allowed to configure the Data-Plane of network nodes (CLI, NETCONF). This can be done through ACP certificates by differentiating them and introduce roles. See Appendix A.9.5.

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

Many details of ACP are designed with security in mind and discussed elsewhere in the document:

IPv6 addresses used by nodes in the ACP are covered as part of the node's domain certificate as described in Section 6.2.2. This allows even verification of ownership of a peer's IPv6 address when using a connection authenticated with the domain certificate.

The ACP acts as a security (and transport) substrate for GRASP inside the ACP such that GRASP is not only protected by attacks from the outside, but also by attacks from compromised inside attackers - by relying not only on hop-by-hop security of ACP secure channels, but adding end-to-end security for those GRASP messages. See Section 6.9.2.

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

ACP provides extensible, auto-configuring hop-by-hop protection of the ACP infrastructure via the negotiation of hop-by-hop secure channel protocols. See Section 6.6.

The ACP is designed to minimize attacks from the outside by minimizing its dependency against any non-ACP (Data-Plane) operations/configuration on a node. See also Section 6.13.2.

In combination with BRSKI, ACP enables a resilient, fully zero-touch network solution for short-lived certificates that can be renewed or re-enrolled even after unintentional expiry (e.g., because of interrupted connectivity). See Appendix A.2.

Because ACP secure channels can be long lived, but certificates used may be short lived, secure channels, for example built via IPsec need to be terminated when peer certificates expire. See Section 6.8.5.

Section 7.2 describes how to implement a routed ACP topology operating on what effectively is a large bridge-domain when using L3/L2 routers that operate at L2 in the Data-Plane. In this case, the ACP is subject to much higher likelihood of attacks by other nodes "stealing" L2 addresses than in the actual routed case. Especially when the bridged network includes non-trusted devices such as hosts. This is a generic issue in L2 LANs. L2/L3 devices often already have some form of "port security" to prohibit this. They rely on NDP or DHCP learning of which port/MAC-address and IPv6 address belong together and block MAC/IPv6 source addresses from wrong ports. This type of function needs to be enabled to prohibit DoS attacks and specifically to protect the ACP. Likewise the GRASP DULL instance needs to ensure that the IPv6 address in the locator-option matches the source IPv6 address of the DULL GRASP packet.

12. IANA Considerations

This document defines the "Autonomic Control Plane".

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

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

The IANA is requested to register the value "AN_ACP" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.4.

The IANA is requested to register the value "SRV.est" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 6.2.5.

Explanation: This document chooses the initially strange looking format "SRV.<service-name>" because these objective names would be in line with potential future simplification of the GRASP objective registry. Today, every name in the GRASP objective registry needs to be explicitly allocated with IANA. In the future, this type of objective names could be considered to be automatically registered in that registry for the same service for which a <service-name> is registered according to [RFC6335]. This explanation is solely informational and has no impact on the requested registration.

The IANA is requested to create an ACP Parameter Registry with currently one registry table - the "ACP Address Type" table.

"ACP Address Type" Table. The value in this table are numeric values 0...3 paired with a name (string). Future values MUST be assigned using the Standards Action policy defined by [RFC8126]. The following initial values are assigned by this document:

0: ACP Zone Addressing Sub-Scheme (ACP RFC Section 6.11.3)

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

13. Acknowledgements

This work originated from an Autonomic Networking project at Cisco Systems, which started in early 2010. Many people contributed to this project and the idea of the Autonomic Control Plane, amongst which (in alphabetical order): Ignas Bagdonas, Parag Bhide, Balaji BL, Alex Clemm, Yves Hertoghs, Bruno Klauser, Max Pritikin, Michael Richardson, Ravi Kumar Vadapalli.

Special thanks to Brian Carpenter, Elwyn Davies, Joel Halpern and Sheng Jiang for their thorough reviews.

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

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

14. Contributors

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

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

Email: brian.e.carpenter@gmail.com

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

Michael C. Richardson
Sandelman Software Works

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

For the RPL technology choices and text.

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

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

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

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

15.1. Summary of changes since entering IESG review

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

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

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

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

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

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

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

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

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

15.1.2. BRSKI / ACP registrar related enhancements

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

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

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

6.1.4 (new) Overview of TA required for ACP.

6.1.5.5 Added explanations/requirements for Re-enrollment.

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

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

15.1.3. Normative enhancements since start of IESG review

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

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

6.1.5.3 Added explanations about CRLs.

6.1.5.6 Added explanations of behavior under failing certificates.

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

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

6.6 Added throttling of secure-channel setup attempts.

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

6.12.5 Added recommendation to use IPv6 DAD.

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

15.1.4. Explanatory enhancements since start of IESG review

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

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

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

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

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

6.5 Elaborated step-by-step secure channel setup.

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

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

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

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

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

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

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

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

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

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

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

Comments from Erik Kline:

2. Fine tuned ULA definition.

Comments Michael Richardson / Eric Vyncke.

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

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

Comments/proposals from Ben Kaduk:

Various: Replaced RFC4492 with RFC8422 which is superceding it.

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

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

6.2.5.4. Fine tuning 2 sentences.

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

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

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

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

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

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

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

A.6 - removed.

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

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

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

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

Other.

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

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

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

Comments from Robert Wilton:

Improved several textual nits.

Discuss/Comments from Erik Kline:

Editorial suggestions and nits. Thanks!.

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

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

6.7 Using RFC8415 terminology for exponential backoff parameters.

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

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

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

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

Discuss/Comments from Barry Leiba:

Various editorial nits - thanks.

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

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

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

6.8.4 Limited mentioning of DTLS version requirements to this section.

6.9.2 Removed TLS requirements, they are now in 6.1.

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

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

Discuss/Comments from Ben Kaduk:

Many good editorial improvements - thanks!.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Discuss from Roman Danyliw:

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

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

Other:

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

Added contributors section.

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

IESG review Roman Danyliw:

6. Requested additional text elaborating misconfiguration plus attack vectors.

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

6.7.2 reworded text about additional secure channel protocol requirements.

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

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

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

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

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

8.2.1. Fixed ABNF 'or' syntax line.

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

10.2.1. Added explanations for attacks from impaired ACP nodes.

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

11. Added note about need for new protocols inside ACP to use end-to-end authentication.

11. Rewrote paragraph about operator mistakes so as to be actionable. Operators must not make mistakes - but ACP minimizes the mistakes they can make.

ACP domain certificate -> ACP certificate.

Various other cosmetic edits (thanks!) and typo fixes (sorry for not running full spell check for every version. Will definitely do before RFC editor).

Other:

6.12.5.2.1./6.12.5.2.2. Added text explaining link breakage wrt. RTL (came about re-analyzing behavior after question about hop count).

Removed now unnecessary references for earlier rrc822Name otherName choice.

15.5. draft-ietf-anima-autonomic-control-plane-27

Too many revisions with too many fixes. Lets do a one-word change revision for a change now if it helps to accelerate the review process.

Added "subjectAltName /" to make it unambiguous that AcpNodeName is indeed a SAN (from Russ).

15.6. draft-ietf-anima-autonomic-control-plane-26

Russ Housley review of -25.

1.1 Explicit reference for TLS 1.2 RFC.

2. Changed term of "ACP Domain Information" to AcpNodeName (ASN.1) / acp-node-name (ABNF), also through rest of document.

2. Improved CA behavior definition. changed IDevID/LDevID to IDevID/LDevID certificate to be more unambiguous.

2. Changed definition of root CA to just refer to how its used in RFC7030 CA root key update, because thats the only thing relevant to ACP.

6.1.1 Moved ECDH requirement to end of text as it was not related to the subject of the initial paragrap. Likewise reference to CABFORUM.

6.1.1 Reduced cert key requirements to only be MUST for certs with 2048 RSA public key and P-256 curves. Reduced longer keys to SHOULD.

6.1.2 Changed text for conversion from rfc822Name to otherName / AcpNode, removed all the explanations of benefits coming with rfc822Name *sob* *sob* *sob*.

6.1.2.1 New ASN.1 definition for otherName / AcpNodeName.

6.1.3 Fixed up text. re the handling of missing connectivity for CRLDP / OCSP.

6.1.4 Fixed up text re. inability to use public CA to situation with otherName / AcpNodeName (no more ACME rfc822Name validation for us *sob* *sob* *sob*).

12. Added ASN.1 registration requests to IANA section.

Appenices. Minor changes for rfc822Name to otherName change.

Various minor verbal fixes/enhancements.

15.7. draft-ietf-anima-autonomic-control-plane-25

Crypto parameter discuss from Valery Smyslov and Paul Wouters and resulting changes.

6.7.2 Moved Michael Richardson suggested diagnostic of signaling TA from IPsec section to this general requirements section and added explanation how this may be inappropriate if TA payload is considered secret by TA owner.

6.7.3.1 Added traffic selectors for native IPsec. Improved text explanation.

6.7.3.1.2 removed misleading text about signaling TA when using intermediate certs.

6.7.3.1.2 Removed requirement for 'PKCS #7 wrapped X.509 certificate' requirement on request of Valery Smyslov as it is not defined in RFC7296 and there are enough options mandated in RFC7296. Replaced with just informative text to educate readers who are not IPsec experts what the mandatory option in RFC7296 is that allows to signal certificates.

6.7.3.1.2 Added SHOULD requirement how to deal with CERTREQ so that 6.7.2 requirement for TA diagnostics will work in IKEv2 (ignoring CERTREQ is permitted by IKEv2). Added explanation how this will result in TA cert diagnostics.

6.7.3.1.2 Added requirement for IKEv2 to operate on link-local addresses for ACP so as to assume ACT cert as the only possible authenticator - to avoid potentially failing section from multiple available certs on a router.

6.7.3.1.2 fixed PKIX- style OID to ASN.1 object AlgorithmIdentifier (Paul).

6.7.3.2 Added IPsec traffic selectors for IPsec with GRE.

6.7.5 Added notion that IPsec/GRE MAY be preferred over IPsec/native. Luckily IPsec/native uses tunneling, whereas IPsec/GRE uses transport mode, and there is a long discuss whether it is permitted to even build IPsec connectings that only support transports instead of always being able to fall back to tunnel mode. Added explanatory paragraph why ACP nodes may prefer GRE over native (wonder how that was missing..).

9.1.1 Added section to explain need for secure channel peer diagnostics via signaling of TA. Four examples given.

Paul Wouters mentioned that ipkcs7 had to be used in some interop cases with windows CA, but that is an issue of ACP Registrar having to convert into PKCS#7 to talk to a windows CA, and this spec is not concerned with that, except to know that it is feasible, so not mentioned in text anywhere, just tracking discussion here in changelog.

Michael Richardson:

3.1.3 Added point in support of rfc822address that CA may not support to sign certificates with new attributes (such as new otherName).

Michael Richardson/Brian Carpenter fix:

6.1.5.1/6.3 Fixed GRASP examples.

Joe Halpern review:

1. Enhanced introduction text for in-band and of out-of-band, explaining how ACP is an in-band network aiming to achieve all possible benefits of an out-of-band network.

1. Comprehensive explanation for term Data-Plane as it is only logically following pre-established terminology on a fully autonomic node, when used for existing nodes augmented with ACP, Data-Plane has more functionality than usually associated with the term.

2. Removed explanatory text for Data-Plane, referring to section 1.

2. Reduced explanation in definition of in-band (management/signaling), out-of-band-signaling, now pointing to section 1.

5. Rewrote a lot of the steps (overview) as this text was not reviewed for long time. Added references to normative section for each step to hopefully avoid feedback of not explaining terms used (really not possible to give good summary without using forward references).

2. Separate out-of-band-management definition from virtual out-of-band-management definition (later one for ACP).

2. Added definitions for RPI and RPL.

6.1.1. added note about end-to-end authentication to distinguish channel security from overall ACP security model.

6.5 Fixed bugs in channel selection signaling step description (Alice vs. Bob).

6.7.1 Removed redundant channel selection explanation.

6.10.3 remove locator/identifier terminology from zone addressing scheme description (unnecessary), removed explanations (now in 9.4), simplified text, clarified requirement for Node-ID to be unique, recommend to use primarily zone 0.

6.10.3.1 Removed. Included a lot of insufficient suggestions for future standards extensions, most of it was wrong or would need to be revisited by WG anyhow. Idea now (just here for comment): Announce via GRASP Zone-ID (e.g. from per-zone edge-node/registrar) into a zone of the ACP so all nodes supporting the scheme can automatically self-allocate the Zone-ID.

6.11.1.1 (RPL overview), eliminated redundant text.

6.11.1.1.1 New subsection to better structure overview.

6.11.1.1.2 New subsection to better group overview, replaced TTL explanation (just the symptom) with hopefully better reconvergence text (intent of the profile) for the ACP RPL profile.

6.11.1.1.6 Added text to explain simple choice for rank_factor.

6.11.1.1.13 moved explanation for RPI up into 6.11.1.1.

6.12.5.1 rewrote section for ACP Loopback Interface.

9.4 New informative/informational section for partial or incremental adoption of ACP to help understand why there is the Zone interface sub-scheme, and how to use it.

Unrelated fixes:

Ask to RFC editor to add most important abbreviations to RFC editor abbreviation list.

6.10.2 changed names in ACP addressing scheme table to be less suggestive of use.

Russ Hously review:

2. Fixed definition of "Enrollment", "Trust Anchor", "CA", and "root CA". Changed "Certificate Authority" to "Certification Authority" throughout the document (correct term according to X.509).

6.1 Fixed explanation of mutual ACP trust.

6.1.1 s/X509/X509v3/.

6.1.2 created bulleted lists for explanations and justifications for choices of ACP certificate encoding. No semantic changes, just to make it easier to refer to the points in discussions (rfcdiff seems to have a bug showing text differences due to formatting changes).

6.1.3 Moved content of rule #1 into previous rule #2 because certification chain validation does imply validation of lifetime. numbers of all rules reduced by 1, changed hopefully all references to the rule numbers in the document.

Rule #3, Hopefully fixed linguistic problem self-contradiction of MUST by lower casing MUST in the explanation part and rewriting the condition when this is not applicable.

6.1.4 Replaced redundant term "Trust Point" (TP) with Trust Anchor (TA). Replaced throughout document Trust Anchor with abbreviation TA.

Enhanced several sentences/rewrote paragraphs to make explanations clearer.

6.6 Added explanation how ACP nodes must throttle their attempts for connection making purely on the result of their own connection attempts, not based on those connections where they are responder.

15.8. draft-ietf-anima-autonomic-control-plane-24

Leftover from -23 review by Eric Vyncke:

Swapping sections 9 and 10, section 9 was meant to be at end of document and summarize. Its not meant to be misinterpreted as introducing any new information. This did happen because section 10 was added after section 9.

15.9. draft-ietf-anima-autonomic-control-plane-23

Note: big rfcdiff of TOC is an rfcdiff bug, changes really minimal.

Review of IPsec security with Mcr and ipsec mailing list.

6.7.1 - new section: Moved general considerations for secure channel protocols here, refined them.

6.7.2 - new section: Moved common requirements for secure channel protocols here, refined them.

6.7.3.1.1. - improved requirements text related to RFC8221, better explanations re. HW acceleration issues.

6.7.3.1.2. - improved requirements text related to RFC8247, (some requirements still discussed to be redundant, will be finalized in next weeks.

Eric Vyncke review of -21:

Only noting most important changes, long list of smaller text/readability enhancements.

2. - New explanation of "normative", "informational" section title tags. alphabetic reordering of terms, refined definitions for CA, CRL. root CA.

6.1.1. - explanation when IDevID parameters may be copied into LDevID.

6.1.2. - Fixed hex digits in ACP domain information to lower case.

6.1.3.1. - New section on Realtime clock and Time Validation.

6.3 - Added explanation that DTLS means \geq version 1.2 (not only 1.2).

6.7 - New text in this main section explaining relationship of ACP secure channels and ACP virtual interfaces - with forward references to virtual interface section.

6.8.2 - reordered text and picture, no text change.

6.10.7.2 - describe first how Registrar-ID can be allocated for all type of registrars, then refined text for how to potentially use MAC addresses on physical registrars.

6.11.1.1 - Added text how this profile does not use Data-Plane artefacts (RPI) because hardware forwarding. This was previously hidden only later in the text.

6.11.1.13. - Rewrote RPL Data-Plane artefact text. Provide decoder ring for abbreviations and all relevant RFCs.

6.12.5.2. - Added more explicit text that secure channels are mapped into virtual interfaces, moved different type of interfaces used by ACP into separate subsections to be able to refer to them.

7.2 - Rewrote/refined text for ACP on L2, prior text was confusing and did not well explain why ACP for L2/L3 switches can be implemented without any L2 (HW) changes. Also missing explanation of only running GRASP untagged when VLANs are used.

8.1.1 - Added requirement for ACP Edge nodes to allow configurable filtering of IPv6 RPI headers.

11. - (security section). Moved explanation of address stealing from 7.2 to here.

15.10. draft-ietf-anima-autonomic-control-plane-22

Ben Kaduk review of -21:

RFC822 encoding of ACP domain information:

6.1.2 rewrote text for explaining / justifying use of rfc822name as identifier for node CP in certificate (was discussed in thread, but badly written in prior versions).

6.1.2 Changed EBNF syntax to use "+" after rfcSELF because that is the known primary name to extensions separator in many email systems ("." was wrong in prior versions).

6.1.2 Rewrote/improved explanations for use of rfc822name field to explain better why it is PKIX compliant and the right thing to do.

Crypto parameters for IPsec:

6.1 - Added explanation of why manual keying for ACP is not feasible for ACP. Surprisingly, that text did not exist. Referred to by IPsec text (6.7.1), but here is the right place to describe the reasoning.

6.1.2 - Small textual refinement referring to requirements to authenticate peers (for the special cases of empty or '0' ACP address in ACP domain information field).

6.3 - To better justify Bens proposed change of secure channel protocol being IPsec vs. GRASP objective being IKEv2, better explained how protocol indicated in GRASP objective-value is name of protocol used to negotiate secure channel, use example of IKEv2 to negotiate IPsec.

6.7.1 - refinemenet similar to 6.3.

- moved new paragraph from Bens pull request up from 6.7.1.1 to 6.7.1 as it equally applies to GRE encapped IPsec (looks nicer one level up).

- created subsections 6.7.1.1 (IPsec/ESP) / 6.7.1.2 (IKEv2) to clearer distinguish between these two requirements blocks.

- Refined the text in these two sections to hopefully be a good answer to Valery's concern of not randomly mocking with existing requirements docs (rfc8247 / rfc8221).

6.7.1.1.1 - IPsec/ESP requirements section:

- MUST support rfc8221 mandatory EXCEPT for the superceeding requirements in this section. Previously, this was not quite clear from the text.

- Hopefully persuasive explanations about the requirements levels for ENCR_AES_GCM_16, ENCR_AES_CBC, ENCR_AES_CCM_8 and ENCR_CHACHA20_POLY1305: Restructured text for why not ENCR_AES_CBC (was in prior version, just not well structured), added new explanations for ENCR_AES_CCM_8 and ENCR_CHACHA20_POLY130.

- In simple terms, requirements for ENCR_AES_CBC, ENCR_AES_CCM_8, ENCR_CHACHA20 are SHOULD when they are implementable with equal or faster performance than ENCR_AES_GCM_16.

- Removed text about "additional rfc8221" requirements MAY be used. Now the logic is that all other requirements apply. Hopefully we have written enough so that we prohibited downgrades.

6.7.1.1.2 - RFC8247 requirements:

- Added mandate to support rfc8247, added explanation that there is no "stripping down" requirement, just additional stronger requirements to mandate correct use of ACP certificates during authentication.

- refined text on identifying ACP by IPv6 address to be clearer: Identifying in the context of IKEv2 and cases for '0' in ACP domain information.

- removed last two paragraphs about relationship to rfc8247, as this is now written in first paragraph of the section.

End of Ben Kaduk review related fixes.

Other:

Forgot to update example of ACP domain information to use capitalized hex-digits as required by HEXDIG used.

Added reference to RFC8316 (AN use-cases) to beginning of section 3 (ACP use cases).

Small Enhanced IPsec parameters description / requirements fixes (from Michael Richardson).

16. Normative References

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

Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-bootstrapping-keyinfra-43, 7 August 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-bootstrapping-keyinfra-43.txt>>.

[I-D.ietf-anima-grasp]

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

[IKEV2IANA]

IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <<https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml>>.

- [RFC1034] Mockapetris, P.V., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6551] Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N., and D. Barthel, "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks", RFC 6551, DOI 10.17487/RFC6551, March 2012, <<https://www.rfc-editor.org/info/rfc6551>>.
- [RFC6552] Thubert, P., Ed., "Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6552, DOI 10.17487/RFC6552, March 2012, <<https://www.rfc-editor.org/info/rfc6552>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<https://www.rfc-editor.org/info/rfc6553>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.

- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7676] Pignataro, C., Bonica, R., and S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC 7676, DOI 10.17487/RFC7676, October 2015, <<https://www.rfc-editor.org/info/rfc7676>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.
- [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <<https://www.rfc-editor.org/info/rfc8247>>.
- [RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <<https://www.rfc-editor.org/info/rfc8422>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

17. Informative References

- [ACPDRAFT] Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-30, <<https://tools.ietf.org/html/draft-ietf-anima-autonomic-control-plane-30.pdf>>. [RFC-Editor: Please remove this complete reference from the RFC] Refer to the IETF working group draft for the few sections removed from this document for various reasons. They capture the state of discussion about unresolved issues that may need to be revisited in future work.
- [AR8021] Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [CABFORUM] CA/Browser Forum, "Certificate Contents for Baseline SSL", November 2019, <<https://cabforum.org/baseline-requirements-certificate-contents/>>.
- [FCC] FCC, "FCC STAFF REPORT ON NATIONWIDE T-MOBILE NETWORK OUTAGE ON JUNE 15, 2020 (PS Docket No. 20-183)", 2020, <<https://docs.fcc.gov/public/attachments/DOC-367699A1.docx>>. The FCC's Public Safety and Homeland Security Bureau issues a report on a nationwide T-Mobile outage that occurred on June 15, 2020. Action by: Public Safety and Homeland Security Bureau.
- [I-D.eckert-anima-noc-autoconfig]
Eckert, T., "Autoconfiguration of NOC services in ACP networks via GRASP", Work in Progress, Internet-Draft, draft-eckert-anima-noc-autoconfig-00, 2 July 2018, <<http://www.ietf.org/internet-drafts/draft-eckert-anima-noc-autoconfig-00.txt>>.
- [I-D.ietf-acme-star]
Sheffer, Y., Lopez, D., Dios, O., Pastor, A., and T. Fossati, "Support for Short-Term, Automatically-Renewed (STAR) Certificates in Automated Certificate Management Environment (ACME)", Work in Progress, Internet-Draft, draft-ietf-acme-star-11, 24 October 2019, <<http://www.ietf.org/internet-drafts/draft-ietf-acme-star-11.txt>>.
- [I-D.ietf-anima-prefix-management]
Jiang, S., Du, Z., Carpenter, B., and Q. Sun, "Autonomic IPv6 Edge Prefix Management in Large-scale Networks", Work in Progress, Internet-Draft, draft-ietf-anima-prefix-

management-07, 18 December 2017, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-prefix-management-07.txt>>.

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

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

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

Richardson, M., "ROLL Applicability Statement Template", Work in Progress, Internet-Draft, draft-ietf-roll-applicability-template-09, 3 May 2016, <<http://www.ietf.org/internet-drafts/draft-ietf-roll-applicability-template-09.txt>>.

[I-D.ietf-tls-dtls13]

Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-38, 29 May 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-tls-dtls13-38.txt>>.

[IEEE-1588-2008]

IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", December 2008, <<http://standards.ieee.org/findstds/standard/1588-2008.html>>.

[IEEE-802.1X]

Group, W. -. H. L. L. P. W., "IEEE Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control", February 2010, <<http://standards.ieee.org/findstds/standard/802.1X-2010.html>>.

[LLDP]

Group, W. -. H. L. L. P. W., "IEEE Standard for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery", June 2016, <<https://standards.ieee.org/findstds/standard/802.1AB-2016.html>>.

- [MACSEC] Group, W. - H. L. L. P. W., "IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security", June 2006, <<https://standards.ieee.org/findstds/standard/802.1AE-2006.html>>.
- [RFC1112] Deering, S.E., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <<https://www.rfc-editor.org/info/rfc1112>>.
- [RFC1492] Finseth, C., "An Access Control Protocol, Sometimes Called TACACS", RFC 1492, DOI 10.17487/RFC1492, July 1993, <<https://www.rfc-editor.org/info/rfc1492>>.
- [RFC1654] Rekhter, Y., Ed. and T. Li, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 1654, DOI 10.17487/RFC1654, July 1994, <<https://www.rfc-editor.org/info/rfc1654>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <<https://www.rfc-editor.org/info/rfc2315>>.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.
- [RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164, DOI 10.17487/RFC3164, August 2001, <<https://www.rfc-editor.org/info/rfc3164>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<https://www.rfc-editor.org/info/rfc3315>>.

- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <<https://www.rfc-editor.org/info/rfc3411>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <<https://www.rfc-editor.org/info/rfc3596>>.
- [RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, DOI 10.17487/RFC3920, October 2004, <<https://www.rfc-editor.org/info/rfc3920>>.
- [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004, <<https://www.rfc-editor.org/info/rfc3954>>.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <<https://www.rfc-editor.org/info/rfc4007>>.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <<https://www.rfc-editor.org/info/rfc4429>>.
- [RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <<https://www.rfc-editor.org/info/rfc4541>>.

- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <<https://www.rfc-editor.org/info/rfc4604>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC4610] Farinacci, D. and Y. Cai, "Anycast-RP Using Protocol Independent Multicast (PIM)", RFC 4610, DOI 10.17487/RFC4610, August 2006, <<https://www.rfc-editor.org/info/rfc4610>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", RFC 4985, DOI 10.17487/RFC4985, August 2007, <<https://www.rfc-editor.org/info/rfc4985>>.
- [RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, DOI 10.17487/RFC5790, February 2010, <<https://www.rfc-editor.org/info/rfc5790>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)", RFC 5912, DOI 10.17487/RFC5912, June 2010, <<https://www.rfc-editor.org/info/rfc5912>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6402] Schaad, J., "Certificate Management over CMS (CMC) Updates", RFC 6402, DOI 10.17487/RFC6402, November 2011, <<https://www.rfc-editor.org/info/rfc6402>>.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<https://www.rfc-editor.org/info/rfc6554>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <<https://www.rfc-editor.org/info/rfc6733>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/info/rfc6824>>.

- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [RFC7404] Behringer, M. and E. Vyncke, "Using Only Link-Local Addressing inside an IPv6 Network", RFC 7404, DOI 10.17487/RFC7404, November 2014, <<https://www.rfc-editor.org/info/rfc7404>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.
- [RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, June 2015, <<https://www.rfc-editor.org/info/rfc7576>>.
- [RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <<https://www.rfc-editor.org/info/rfc7585>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.

- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8316] Nobre, J., Granville, L., Clemm, A., and A. Gonzalez Prieto, "Autonomic Networking Use Case for Distributed Detection of Service Level Agreement (SLA) Violations", RFC 8316, DOI 10.17487/RFC8316, February 2018, <<https://www.rfc-editor.org/info/rfc8316>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <<https://www.rfc-editor.org/info/rfc8368>>.
- [RFC8398] Melnikov, A., Ed. and W. Chuang, Ed., "Internationalized Email Addresses in X.509 Certificates", RFC 8398, DOI 10.17487/RFC8398, May 2018, <<https://www.rfc-editor.org/info/rfc8398>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

- [RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <<https://www.rfc-editor.org/info/rfc8572>>.
- [X.509] International Telecommunication Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, ISO/IEC 9594-8, October 2016, <<https://www.itu.int/rec/T-REC-X.509>>.
- [X.520] International Telecommunication Union, "Information technology - Open Systems Interconnection - The Directory: Selected attribute types", ITU-T Recommendation X.520, ISO/IEC 9594-6, October 2016, <<https://www.itu.int/rec/T-REC-X.520>>.

Appendix A. Background and Futures (Informative)

The following sections discuss additional background information about aspects of the normative parts of this document or associated mechanisms such as BRSKI (such as why specific choices were made by the ACP) and they provide discussion about possible future variations of the ACP.

A.1. ACP Address Space Schemes

This document defines the Zone, Vlong and Manual sub address schemes primarily to support address prefix assignment via distributed, potentially uncoordinated ACP registrars as defined in Section 6.11.7. This costs 48/46-bit identifier so that these ACP registrar can assign non-conflicting address prefixes. This design does not leave enough bits to simultaneously support a large number of nodes (Node-ID) plus a large prefix of local addresses for every node plus a large enough set of bits to identify a routing Zone. In result, Zone, Vlong 8/16 attempt to support all features, but via separate prefixes.

In networks that always expect to rely on a centralized PMS as described above (Section 9.2.5), the 48/46-bits for the Registrar-ID could be saved. Such variations of the ACP addressing mechanisms could be introduced through future work in different ways. If a new otherName was introduced, incompatible ACP variations could be created where every design aspect of the ACP could be changed. Including all addressing choices. If instead a new addressing sub-type would be defined, it could be a backward compatible extension of this ACP specification. Information such as the size of a zone-prefix and the length of the prefix assigned to the ACP node itself could be encoded via the extension field of the acp-node-name.

Note that an explicitly defined "Manual" addressing sub-scheme is always beneficial to provide an easy way for ACP nodes to prohibit incorrect manual configuration of any non-"Manual" ACP address spaces and therefore ensure that "Manual" operations will never impact correct routing for any non-"Manual" ACP addresses assigned via ACP certificates.

A.2. BRSKI Bootstrap (ANI)

BRSKI describes how nodes with an IDevID certificate can securely and zero-touch enroll with an LDevID certificate to support the ACP. BRSKI also leverages the ACP to enable zero-touch bootstrap of new nodes across networks without any configuration requirements across the transit nodes (e.g., no DHCP/DNS forwarding/server setup). This includes otherwise not configured networks as described in Section 3.2. Therefore, BRSKI in conjunction with ACP provides for a secure and zero-touch management solution for complete networks. Nodes supporting such an infrastructure (BRSKI and ACP) are called ANI nodes (Autonomic Networking Infrastructure), see [I-D.ietf-anima-reference-model]. Nodes that do not support an IDevID certificate but only an (insecure) vendor specific Unique Device Identifier (UDI) or nodes whose manufacturer does not support a MASA could use some future security reduced version of BRSKI.

When BRSKI is used to provision a domain certificate (which is called enrollment), the BRSKI registrar (acting as an enhanced EST server) must include the otherName / AcpNodeName encoded ACP address and domain name to the enrolling node (called pledge) via its response to the pledges EST CSR Attribute request that is mandatory in BRSKI.

The Certification Authority in an ACP network must not change the otherName / AcpNodeName in the certificate. The ACP nodes can therefore find their ACP address and domain using this field in the domain certificate, both for themselves, as well as for other nodes.

The use of BRSKI in conjunction with the ACP can also help to further simplify maintenance and renewal of domain certificates. Instead of relying on CRL, the lifetime of certificates can be made extremely small, for example in the order of hours. When a node fails to connect to the ACP within its certificate lifetime, it cannot connect to the ACP to renew its certificate across it (using just EST), but it can still renew its certificate as an "enrolled/expired pledge" via the BRSKI bootstrap proxy. This requires only that the BRSKI registrar honors expired domain certificates and that the pledge attempts to perform TLS authentication for BRSKI bootstrap using its expired domain certificate before falling back to attempting to use its IDevID certificate for BRSKI. This mechanism could also render CRLs unnecessary because the BRSKI registrar in conjunction with the CA would not renew revoked certificates - only a "Do-not-renew" list would be necessary on BRSKI registrars/CA.

In the absence of BRSKI or less secure variants thereof, provisioning of certificates may involve one or more touches or non-standardized automation. Node vendors usually support provisioning of certificates into nodes via PKCS#7 (see [RFC2315]) and may support this provisioning through vendor specific models via NETCONF ([RFC6241]). If such nodes also support NETCONF Zero-Touch ([RFC8572]) then this can be combined to zero-touch provisioning of domain certificates into nodes. Unless there are equivalent integration of NETCONF connections across the ACP as there is in BRSKI, this combination would not support zero-touch bootstrap across a not configured network though.

A.3. ACP Neighbor discovery protocol selection

This section discusses why GRASP DULL was chosen as the discovery protocol for L2 adjacent candidate ACP neighbors. The contenders considered where GRASP, mDNS or LLDP.

A.3.1. LLDP

LLDP and Cisco's earlier Cisco Discovery Protocol (CDP) are example of L2 discovery protocols that terminate their messages on L2 ports. If those protocols would be chosen for ACP neighbor discovery, ACP neighbor discovery would therefore also terminate on L2 ports. This would prevent ACP construction over non-ACP capable but LLDP or CDP enabled L2 switches. LLDP has extensions using different MAC addresses and this could have been an option for ACP discovery as well, but the additional required IEEE standardization and definition of a profile for such a modified instance of LLDP seemed to be more work than the benefit of "reusing the existing protocol" LLDP for this very simple purpose.

A.3.2. mDNS and L2 support

Multicast DNNS (mDNS) [RFC6762] with DNS Service Discovery (DNS-SD) Resource Records (RRs) as defined in [RFC6763] is a key contender as an ACP discovery protocol. because it relies on link-local IP multicast, it does operates at the subnet level, and is also found in L2 switches. The authors of this document are not aware of mDNS implementation that terminate their mDNS messages on L2 ports instead of the subnet level. If mDNS was used as the ACP discovery mechanism on an ACP capable (L3)/L2 switch as outlined in Section 7, then this would be necessary to implement. It is likely that termination of mDNS messages could only be applied to all mDNS messages from such a port, which would then make it necessary to software forward any non-ACP related mDNS messages to maintain prior non-ACP mDNS functionality. Adding support for ACP into such L2 switches with mDNS could therefore create regression problems for prior mDNS functionality on those nodes. With low performance of software forwarding in many L2 switches, this could also make the ACP risky to support on such L2 switches.

A.3.3. Why DULL GRASP

LLDP was not considered because of the above mentioned issues. mDNS was not selected because of the above L2 mDNS considerations and because of the following additional points:

If mDNS was not already existing in a node, it would be more work to implement than DULL GRASP, and if an existing implementation of mDNS was used, it would likely be more code space than a separate implementation of DULL GRASP or a shared implementation of DULL GRASP and GRASP in the ACP.

A.4. Choice of routing protocol (RPL)

This section motivates why RPL - "IPv6 Routing Protocol for Low-Power and Lossy Networks ([RFC6550] was chosen as the default (and in this specification only) routing protocol for the ACP. The choice and above explained profile was derived from a pre-standard implementation of ACP that was successfully deployed in operational networks.

Requirements for routing in the ACP are:

- * Self-management: The ACP must build automatically, without human intervention. Therefore, routing protocol must also work completely automatically. RPL is a simple, self-managing protocol, which does not require zones or areas; it is also self-configuring, since configuration is carried as part of the protocol (see Section 6.7.6 of [RFC6550]).
- * Scale: The ACP builds over an entire domain, which could be a large enterprise or service provider network. The routing protocol must therefore support domains of 100,000 nodes or more, ideally without the need for zoning or separation into areas. RPL has this scale property. This is based on extensive use of default routing.
- * Low resource consumption: The ACP supports traditional network infrastructure, thus runs in addition to traditional protocols. The ACP, and specifically the routing protocol must have low resource consumption both in terms of memory and CPU requirements. Specifically, at edge nodes, where memory and CPU are scarce, consumption should be minimal. RPL builds a DODAG, where the main resource consumption is at the root of the DODAG. The closer to the edge of the network, the less state needs to be maintained. This adapts nicely to the typical network design. Also, all changes below a common parent node are kept below that parent node.
- * Support for unstructured address space: In the Autonomic Networking Infrastructure, node addresses are identifiers, and may not be assigned in a topological way. Also, nodes may move topologically, without changing their address. Therefore, the routing protocol must support completely unstructured address space. RPL is specifically made for mobile ad-hoc networks, with no assumptions on topologically aligned addressing.
- * Modularity: To keep the initial implementation small, yet allow later for more complex methods, it is highly desirable that the routing protocol has a simple base functionality, but can import new functional modules if needed. RPL has this property with the concept of "objective function", which is a plugin to modify routing behavior.
- * Extensibility: Since the Autonomic Networking Infrastructure is a new concept, it is likely that changes in the way of operation will happen over time. RPL allows for new objective functions to be introduced later, which allow changes to the way the routing protocol creates the DAGs.
- * Multi-topology support: It may become necessary in the future to support more than one DODAG for different purposes, using different objective functions. RPL allow for the creation of several parallel DODAGs, should this be required. This could be used to create different topologies to reach different roots.

- * No need for path optimization: RPL does not necessarily compute the optimal path between any two nodes. However, the ACP does not require this today, since it carries mainly non-delay-sensitive feedback loops. It is possible that different optimization schemes become necessary in the future, but RPL can be expanded (see point "Extensibility" above).

A.5. ACP Information Distribution and multicast

IP multicast is not used by the ACP because the ANI (Autonomic Networking Infrastructure) itself does not require IP multicast but only service announcement/discovery. Using IP multicast for that would have made it necessary to develop a zero-touch auto configuring solution for ASM (Any Source Multicast - the original form of IP multicast defined in [RFC1112]), which would be quite complex and difficult to justify. One aspect of complexity where no attempt at a solution has been described in IETF documents is the automatic-selection of routers that should be PIM Sparse Mode (PIM-SM) Rendezvous Points (RPs) (see [RFC7761]). The other aspects of complexity are the implementation of MLD ([RFC4604]), PIM-SM and Anycast-RP (see [RFC4610]). If those implementations already exist in a product, then they would be very likely tied to accelerated forwarding which consumes hardware resources, and that in return is difficult to justify as a cost of performing only service discovery.

Some future ASA may need high performance in-network data replication. That is the case when the use of IP multicast is justified. Such an ASA can then use service discovery from ACP GRASP, and then they do not need ASM but only SSM (Source Specific Multicast, see [RFC4607]) for the IP multicast replication. SSM itself can simply be enabled in the Data-Plane (or even in an update to the ACP) without any other configuration than just enabling it on all nodes and only requires a simpler version of MLD (see [RFC5790]).

LSP (Link State Protocol) based IGP routing protocols typically have a mechanism to flood information, and such a mechanism could be used to flood GRASP objectives by defining them to be information of that IGP. This would be a possible optimization in future variations of the ACP that do use an LSP routing protocol. Note though that such a mechanism would not work easily for GRASP M_DISCOVERY messages which are intelligently (constrained) flooded not across the whole ACP, but only up to a node where a responder is found. We do expect that many future services in ASA will have only few consuming ASA, and for those cases, M_DISCOVERY is the more efficient method than flooding across the whole domain.

Because the ACP uses RPL, one desirable future extension is to use RPLs existing notion of DODAG, which are loop-free distribution trees, to make GRASP flooding more efficient both for M_FLOOD and M_DISCOVERY. See Section 6.13.5 how this will be specifically beneficial when using NBMA interfaces. This is not currently specified in this document because it is not quite clear yet what exactly the implications are to make GRASP flooding depend on RPL DODAG convergence and how difficult it would be to let GRASP flooding access the DODAG information.

A.6. CAs, domains and routing subdomains

There is a wide range of setting up different ACP solution by appropriately using CAs and the domain and rsub elements in the acp-node-name in the domain certificate. We summarize these options here as they have been explained in different parts of the document in before and discuss possible and desirable extensions:

An ACP domain is the set of all ACP nodes that can authenticate each other as belonging to the same ACP network using the ACP domain membership check (Section 6.2.3). GRASP inside the ACP is run across all transitively connected ACP nodes in a domain.

The rsub element in the acp-node-name permits the use of addresses from different ULA prefixes. One use case is to create multiple physical networks that initially may be separated with one ACP domain but different routing subdomains, so that all nodes can mutual trust their ACP certificates (not depending on rsub) and so that they could connect later together into a contiguous ACP network.

One instance of such a use case is an ACP for regions interconnected via a non-ACP enabled core, for example due to the absence of product support for ACP on the core nodes. ACP connect configurations as defined in this document can be used to extend and interconnect those ACP islands to the NOC and merge them into a single ACP when later that product support gap is closed.

Note that RPL scales very well. It is not necessary to use multiple routing subdomains to scale ACP domains in a way that would be required if other routing protocols 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 acp-node-name. These domain elements can be arbitrary, including subdomains of one another: Domains "example.com" and "research.example.com" are separate domains if both are domain elements in the acp-node-name of certificates.

It is not necessary to have a separate CA for different ACP domains: an operator can use a single CA to sign certificates for multiple ACP domains that are not allowed to connect to each other because the checks for ACP adjacencies includes comparison of the domain part.

If multiple independent networks choose the same domain name but had their own CA, these would not form a single ACP domain because of CA mismatch. Therefore, there is no problem in choosing domain names that are potentially also used by others. Nevertheless it is highly recommended to use domain names that one can have high probability to be unique. It is recommended to use domain names that start with a DNS domain names owned by the assigning organization and unique within it. For example, "acp.example.com" if you own "example.com".

A.7. Intent for the ACP

Intent is the architecture component of autonomic networks according to [I-D.ietf-anima-reference-model] that allows operators to issue policies to the network. Its applicability for use is quite flexible and freeform, with potential applications including policies flooded across ACP GRASP and interpreted on every ACP node.

One concern for future definitions of Intent solutions is the problem of circular dependencies when expressing Intent policies about the ACP itself.

For example, Intent could indicate the desire to build an ACP across all domains that have a common parent domain (without relying on the rsub/routing-subdomain solution defined in this document). For example, ACP nodes with domain "example.com", "access.example.com", "core.example.com" and "city.core.example.com" should all establish one single ACP.

If each domain has its own source of Intent, then the Intent would simply have to allow adding the peer domains TA and domain names to the parameters for the ACP domain membership check (Section 6.2.3) so that nodes from those other domains are accepted as ACP peers.

If this Intent was to be originated only from one domain, it could likely not be made to work because the other domains will not build any ACP connection amongst each other, whether they use the same or different CA due to the ACP domain membership check.

If the domains use the same CA one could change the ACP setup to permit for the ACP to be established between two ACP nodes with different `acp-domain-names`, but only for the purpose of disseminating limited information, such as Intent, but not to set up full ACP connectivity, specifically not RPL routing and passing of arbitrary GRASP information. Unless the Intent policies permit this to happen across domain boundaries.

This type of approach where the ACP first allows Intent to operate and only then sets up the rest of ACP connectivity based on Intent policy could also be used to enable Intent policies that would limit functionality across the ACP inside a domain, as long as no policy would disturb the distribution of Intent. For example, to limit reachability across the ACP to certain type of nodes or locations of nodes.

A.8. Adopting ACP concepts for other environments

The ACP as specified in this document is very explicit about the choice of options to allow interoperable implementations. The choices made may not be the best for all environments, but the concepts used by the ACP can be used to build derived solutions:

The ACP specifies the use of ULA and deriving its prefix from the domain name so that no address allocation is required to deploy the ACP. The ACP will equally work not using ULA but any other /48 IPv6 prefix. This prefix could simply be a configuration of the ACP registrars (for example when using BRSKI) to enroll the domain certificates - instead of the ACP registrar deriving the /48 ULA prefix from the AN domain name.

Some solutions may already have an auto-addressing scheme, for example derived from existing unique device identifiers (e.g., MAC addresses). In those cases it may not be desirable to assign addresses to devices via the ACP address information field in the way described in this document. The certificate may simply serve to identify the ACP domain, and the address field could be omitted. The only fix required in the remaining way the ACP operate is to define another element in the domain certificate for the two peers to decide who is the Decider and who is the Follower during secure channel building. Note though that future work may leverage the `acp address` to authenticate "ownership" of the address by the device. If the address used by a device is derived from some pre-existing permanent local ID (such as MAC address), then it would be useful to store that address in the certificate using the format of the access address information field or in a similar way.

The ACP is defined as a separate VRF because it intends to support well managed networks with a wide variety of configurations. Therefore, reliable, configuration-indestructible connectivity cannot be achieved from the Data-Plane itself. In solutions where all transit connectivity impacting functions are fully automated (including security), indestructible and resilient, it would be possible to eliminate the need for the ACP to be a separate VRF. Consider the most simple example system in which there is no separate Data-Plane, but the ACP is the Data-Plane. Add BRSKI, and it becomes a fully autonomic network - except that it does not support automatic addressing for user equipment. This gap can then be closed for example by adding a solution derived from [I-D.ietf-anima-prefix-management].

TCP/TLS as the protocols to provide reliability and security to GRASP in the ACP may not be the preferred choice in constrained networks. For example, CoAP/DTLS (Constrained Application Protocol) may be preferred where they are already used, allowing to reduce the additional code space footprint for the ACP on those devices. Hop-by-hop reliability for ACP GRASP messages could be made to support protocols like DTLS by adding the same type of negotiation as defined in this document for ACP secure channel protocol negotiation. End-to-end GRASP connections can be made to select their transport protocol in future extensions of the ACP meant to better support constrained devices by indicating the supported transport protocols (e.g. TLS/DTLS) via GRASP parameters of the GRASP objective through which the transport endpoint is discovered.

The routing protocol RPL used for the ACP does explicitly not optimize for shortest paths and fastest convergence. Variations of the ACP may want to use a different routing protocol or introduce more advanced RPL profiles.

Variations such as what routing protocol to use, or whether to instantiate an ACP in a VRF or (as suggested above) as the actual Data-Plane, can be automatically chosen in implementations built to support multiple options by deriving them from future parameters in the certificate. Parameters in certificates should be limited to those that would not need to be changed more often than certificates would need to be updated anyhow; Or by ensuring that these parameters can be provisioned before the variation of an ACP is activated in a node. Using BRSKI, this could be done for example as additional follow-up signaling directly after the certificate enrollment, still leveraging the BRSKI TLS connection and therefore not introducing any additional connectivity requirements.

Last but not least, secure channel protocols including their encapsulations are easily added to ACP solutions. ACP hop-by-hop network layer secure channels could also be replaced by end-to-end security plus other means for infrastructure protection. Any future network OAM should always use end-to-end security anyhow and can leverage the domain certificates and is therefore not dependent on security to be provided for by ACP secure channels.

A.9. Further (future) options

A.9.1. Auto-aggregation of routes

Routing in the ACP according to this specification only leverages the standard RPL mechanism of route optimization, e.g. keeping only routes that are not towards the RPL root. This is known to scale to networks with 20,000 or more nodes. There is no auto-aggregation of routes for /48 ULA prefixes (when using rsub in the acp-node-name) and/or Zone-ID based prefixes.

Automatic assignment of Zone-ID and auto-aggregation of routes could be achieved for example by configuring zone-boundaries, announcing via GRASP into the zones the zone parameters (zone-ID and /48 ULA prefix) and auto-aggregating routes on the zone-boundaries. Nodes would assign their Zone-ID and potentially even /48 prefix based on the GRASP announcements.

A.9.2. More options for avoiding IPv6 Data-Plane dependencies

As described in Section 6.13.2, the ACP depends on the Data-Plane to establish IPv6 link-local addressing on interfaces. Using a separate MAC address for the ACP allows to fully isolate the ACP from the Data-Plane in a way that is compatible with this specification. It is also an ideal option when using Single-root input/output virtualization (SR-IOV - see https://en.wikipedia.org/wiki/Single-root_input/output_virtualization) in an implementation to isolate the ACP because different SR-IOV interfaces use different MAC addresses.

When additional MAC address(es) are not available, separation of the ACP could be done at different demux points. The same subnet interface could have a separate IPv6 interface for the ACP and Data-Plane and therefore separate link-local addresses for both, where the ACP interface is non-configurable on the Data-Plane. This too would be compatible with this specification and not impact interoperability.

An option that would require additional specification is to use a different Ethertype from 0x86DD (IPv6) to encapsulate IPv6 packets for the ACP. This would be a similar approach as used for IP

authentication packets in [IEEE-802.1X] which use the Extensible Authentication Protocol over Local Area Network (EAPoL) ethertype (0x88A2).

Note that in the case of ANI nodes, all the above considerations equally apply to the encapsulation of BRSKI packets including GRASP used for BRSKI.

A.9.3. ACP APIs and operational models (YANG)

Future work should define YANG ([RFC7950]) data model and/or node internal APIs to monitor and manage the ACP.

Support for the ACP Adjacency Table (Section 6.3) and ACP GRASP need to be included into such model/API.

A.9.4. RPL enhancements

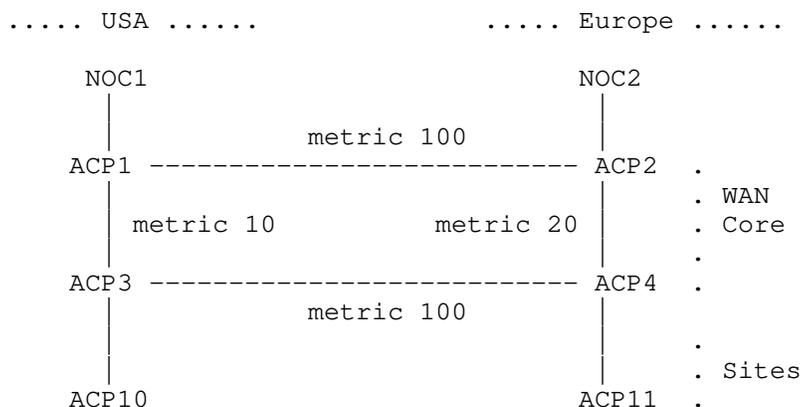


Figure 19: Dual NOC

The profile for RPL specified in this document builds only one spanning-tree path set to a root, typically a registrar in one NOC. In the presence of multiple NOCs, routing toward the non-root NOCs may be suboptimal. Figure 19 shows an extreme example. Assuming that node ACP1 becomes the RPL root, traffic between ACP11 and NOC2 will pass through ACP4-ACP3-ACP1-ACP2 instead of ACP4-ACP2 because the RPL calculated DODAG/routes are shortest paths towards the RPL root.

To overcome these limitations, extensions/modifications to the RPL profile can provide optimality for multiple NOCs. This requires utilizing Data-Plane artifact including IPinIP encap/decap on ACP routers and processing of IPv6 RPI headers. Alternatively, (Src,Dst) routing table entries could be used.

Flooding of ACP GRASP messages can be further constrained and therefore optimized by flooding only via links that are part of the RPL DODAG.

A.9.5. Role assignments

ACP connect is an explicit mechanism to "leak" ACP traffic explicitly (for example in a NOC). It is therefore also a possible security gap when it is easy to enable ACP connect on arbitrary compromised ACP nodes.

One simple solution is to define an extension in the ACP certificates ACP information field indicating the permission for ACP connect to be configured on that ACP node. This could similarly be done to decide whether a node is permitted to be a registrar or not.

Tying the permitted "roles" of an ACP node to the ACP certificate provides fairly strong protection against misconfiguration, but is still subject to code modifications.

Another interesting role to assign to certificates is that of a NOC node. This would allow to limit certain type of connections such as OAM TLS connections to only NOC initiator or responders.

A.9.6. Autonomic L3 transit

In this specification, the ACP can only establish autonomic connectivity across L2 hops and only explicitly configured options to tunnel across L3. Future work should specify mechanisms to automatically tunnel ACP across L3 networks. A hub&spoke option would allow to tunnel across the Internet to a cloud or central instance of the ACP, a peer-to-peer tunneling mechanism could tunnel ACP islands across an L3VPN infrastructure.

A.9.7. Diagnostics

Section 9.1 describes diagnostics options that can be done without changing the external, interoperability affecting characteristics of ACP implementations.

Even better diagnostics of ACP operations is possible with additional signaling extensions, such as:

1. Consider if LLDP should be a recommended functionality for ANI devices to improve diagnostics, and if so, which information elements it should signal (noting that such information is conveyed in an insecure manner). Includes potentially new information elements.
2. In alternative to LLDP, A DULL GRASP diagnostics objective could be defined to carry these information elements.
3. The IDevID certificate of BRSKI pledges should be included in the selected insecure diagnostics option. This may be undesirable when exposure of device information is seen as too much of a security issue (ability to deduce possible attack vectors from device model for example).
4. A richer set of diagnostics information should be made available via the secured ACP channels, using either single-hop GRASP or network wide "topology discovery" mechanisms.

A.9.8. Avoiding and dealing with compromised ACP nodes

Compromised ACP nodes pose the biggest risk to the operations of the network. The most common type of compromise is leakage of credentials to manage/configure the device and the application of malicious configuration including the change of access credentials, but not the change of software. Most of today's networking equipment should have secure boot/software infrastructure anyhow, so attacks that introduce malicious software should be a lot harder.

The most important aspect of security design against these type of attacks is to eliminate password based configuration access methods and instead rely on certificate based credentials handed out only to nodes where it is clear that the private keys cannot leak. This limits unexpected propagation of credentials.

If password based credentials to configure devices still need to be supported, they must not be locally configurable, but only be remotely provisioned or verified (through protocols like RADIUS or Diameter), and there must be no local configuration permitting to change these authentication mechanisms, but ideally they should be autoconfiguring across the ACP. See [I-D.eckert-anima-noc-autoconfig].

Without physical access to the compromised device, attackers with access to configuration should not be able to break the ACP connectivity, even when they can break or otherwise manipulate (spoof) the Data-Plane connectivity through configuration. To achieve this, it is necessary to avoid providing configuration options for the ACP, such as enabling/disabling it on interfaces. For example, there could be an ACP configuration that locks down the current ACP config unless factory reset is done.

With such means, the valid administration has the best chances to maintain access to ACP nodes, discover malicious configuration through ongoing configuration tracking from central locations for example, and to react accordingly.

The primary reaction is withdrawal/change of credentials, terminate malicious existing management sessions and fixing the configuration. Ensuring that management sessions using invalidated credentials are terminated automatically without recourse will likely require new work.

Only when these steps are not feasible would it be necessary to revoke or expire the ACP certificate credentials and consider the node kicked off the network - until the situation can be further rectified, likely requiring direct physical access to the node.

Without extensions, compromised ACP nodes can only be removed from the ACP at the speed of CRL/OCSP information refresh or expiry (and non-removal) of short lived certificates. Future extensions to the ACP could for example use GRASP flooding distribution of triggered updates of CRL/OCSP or explicit removal indication of the compromised nodes domain certificate.

A.9.9. Detecting ACP secure channel downgrade attacks

The following text proposes a mechanism to protect against downgrade attacks without introducing a new specialized UPFRONT GRASP secure channel mechanism. Instead, it relies on running GRASP after establishing a secure channel protocol to verify if the established secure channel option could have been the result of a MITM downgrade attack:

MITM attackers can force downgrade attacks for ACP secure channel selection by filtering/modifying DULL GRASP messages and/or actual secure channel data packets. For example, if at some point in time DTLS traffic could be easier decrypted than traffic of IKEv2, the MITM could filter all IKEv2 packets to force ACP nodes to use DTLS (assuming the ACP nodes in question supported both DTLS and IKEv2).

For cases where such MITM attacks are not capable to inject malicious traffic (but only to decrypt the traffic), a downgrade attack could be discovered after a secure channel connection is established, for example by use of the following type of mechanism:

After the secure channel connection is established, the two ACP peers negotiate via an appropriate (To Be Defined) GRASP negotiation which ACP secure channel protocol should have been selected between them (in the absence of a MITM attacker). This negotiation would have to

signal the DULL GRASP announced ACP secure channel options by each peer followed by an announcement of the preferred secure channel protocol by the ACP peer that is the Decider in the secure channel setup, e.g. the ACP peer that is deciding which secure channel protocol to pick. If that chosen secure channel protocol is different from the one that actually was chosen, then this mismatch is an indication that there is a MITM attacker or other similar issue (firewall prohibiting the use of specific protocols) that caused a non-preferred secure channel protocol to be chosen. This discovery could then result in mitigation options such as logging and ensuing investigations.

Appendix B. Unfinished considerations (To Be Removed From RFC)

[RFC-Editor: This whole appendix B. and its subsections to be removed for the RFC.

This appendix contains unfinished considerations that are removed from the RFC, they are maintained in this draft as a log of the state of discussion and point of reference. Together with this appendix, also the references pointing to it are marked to be removed from the RFC because no consensus could be reached that a self-reference to a draft version of the RFC is an appropriate breadcrumb to point to unfinished considerations.

The authors plan to move these considerations into a new target informational draft, please look for draft-eckert-anima-acp-considerations.

B.1. Considerations for improving secure channel negotiation

Proposed text from Benjamin Kaduk. It is suggested to replace the text of appendix A.6 in previous versions of this draft (up to version 29).

The discovery procedure in this specification for low-level ACP channel support by layer-2 peers involves DULL GRASP and attempting (usually in parallel) to establish all supported channel types, learning the peer ACP address and correspondingly the assignment of Decider and Follower roles, and tearing down all channels other than the one preferred by the Decider. This procedure, in general, becomes resource intensive as the number of possible secure channels grows; even worse, under some threat models, the security of the discovery result is only as strong as the weakest supported secure channel protocol. Furthermore, the unilateral determination of "best" channel type by the Decider does not result in the optimal outcome in all possible scenarios.

This situation is tolerable at present, with only two secure channels (DTLS and IPsec) defined, but long-term agility in the vein of [BCP201] will require the introduction of an alternate discovery/negotiation procedure. While IKEv2 is the IETF standard protocol for negotiating security associations, it currently does not have a defined mechanism flexible enough to negotiate the parameters needed for, e.g., an ACP DTLS channel, let alone for allowing ACP peers to indicate their preference metrics for channel selection. Such a mechanism or mechanisms could be defined, but if ACP agility requires introducing a new channel type, for example MacSec, IKEv2 would again need to be extended in order to negotiate an ACP MacSec association. Making ACP channel agility dependent on updates to IKEv2 is likely to result in obstacles due to different timescales of evolution, since IKEv2 implementations help form the core of Internet-scale security infrastructure and must accordingly be robust and thoroughly tested.

Accordingly, a dedicated ACP channel negotiation mechanism is appropriate as a way to provide long-term algorithm and secure-channel protocol agility. Such a mechanism is not currently defined, but one possible design is as follows. A new DULL GRASP objective is defined to indicate the GRASP-over-TLS channel, which is by definition preferred to other channel types (including DTLS and IPsec). When both peers advertise support for GRASP-over-TLS, GRASP-over-TLS must run to completion before other channel types are attempted. The GRASP-over-TLS channel performs the necessary negotiation by establishing a TLS connection between the peers and using that connection to secure a dedicated GRASP instance for negotiating supported channel types and preference metrics. This provides a rich language for determining what secure channel protocol to use for the ACP link while taking into account the capabilities and preferences of the ACP peers, all protected by the security of the TLS channel.

B.2. ACP address verification

The AcpNodeName of most ACP nodes contains in the acp-address field the primary ACP address to be used by the node for end-to-end connections across ACP secure channels. Nevertheless, there is no verification of an ACP peers address specified in this document. This section explains the current understanding as to why this is not done.

Not all ACP nodes will have an actual IPv6 address in the acp-address field of their AcpNodeName. Those who do not include nodes that do not support ACP secure channels, such as pre-existing NOC equipment that only connects to the ACP via ACP connect interfaces. Likewise, future ACP node type that may want to have their Node-ID not be defined by an ACP registrar, but differently cannot have the ACP

address be provided in their ACP certificate where it would be defined by the registrar. In result, any scheme that would rely on verification of the acp-address in the ACP certificate would only apply to a subset of ACP nodes.

The transport stack network layer address used for ACP secure channels is not the acp-address. For automatically established ACP secure channels, it is a link-local IPv6 address. For explicitly configured ACP secure channels (to reach across non ACP L3 network segments), the address is any IPv4 or IPv6 address routable to that remote destination.

When the acp-address is actually used across the ACP, it can only be verified by a peer when the peer has the certificate of the peer. Unless further higher layer mechanisms are developed on top of the ACP (for example via ASA), the only mechanism to access a peers ACP certificate is for secure connections in which the peers certificates are exchanged and cryptographically verified, e.g. TLS and DTLS. Initially, it is expected that the ACP will carry many legacy network management control connections that unfortunately not end-to-end authenticated but that are solely protected by being carried across the ACP secure channels. ACP address verification therefore cannot be used for such connections without additional higher layer components.

For the remaining (TLS/DTLS) connections for which address verification can be used, the main question is: what additional benefit would address verification provide?

The main value that transport stack network layer address verification could provide for these type of connections would be the discovery of on-path transport proxies. For example, in case of BRSKI, pledges connect to an ACP registrar via an ASA implementing a TCP proxy because the pledge itself has at that point in time no ACP certificate valid to build ACP secure channels and hence needs to rely on such a proxy. This is one example where such a TCP proxy is required and not a form of attack.

In general, on path TCP proxies could be a form of attack, but it stands to reason, that an attacker that manages to enable a malicious TCP proxy could likely equally build a transparent proxy not changing the network layer addresses. Only when the attacker operates off-path would this option not be possible. Such attacks could indeed be possible: An impaired ACP node could announce itself as another service instance for a service whose utilization it wants to attack. It could then attempt to look like a valid server by simply TCP proxying the clients connections to a valid server and then attack the connections passing through it (passive decrypting or passive

fingerprint analysis). But like the BRSKI proxy, this behavior could be perfectly legitimate and not an attack. For example, TCP has in the past often suffered from performance issues across difficult (high capacity, high loss) paths, and TCP proxies where and are being used simply as a tool for isolating such path segments (such as a WAN), and providing caching and local-retransmit of in-transit packets, reducing the effective path segment capacity.

As explained elsewhere in this document already, considerations for these type of attack are therefore outside the scope of the ACP but fundamental to further design of the ASA infrastructure. Beyond distinguishing whether a TCP proxy would be beneficial or malicious, the even more fundamental question is how to determine from a multitude of service announcements which instance is the most trustworthy and functionally best. In the Internet/web, this question is NOT solved inside the network but through off-net human interaction ("trust me, the best search engine is www.<insert-your-personal-recommendation>.com").

B.3. Public CA considerations

Public CAs are outside the scope of this document for the following reasons. This appendix describes the current state of understanding for those interested to consider utilizing public CA for the ACP in the future.

If public CA were to be used to enroll ACP nodes and act as TA, this would require a model in which the public CA would be able to assert the ownership of the information requested in the certificate, especially the AcpNodeName, for example mitigated by the domain registrar(s). Due to the use of a new, ACP unique encoding of the AcpNodeName, there is no mechanism for public CA to do so. More importantly though, isolation between ACPs of disjoint operated ACPs is achieved in the current ACP design through disjoint TA. A public CA is in general based on a single (set of) TA shared across all certificates signed by the CA.

Due to the fact that the ACP domain membership check also validates that a peers domain name in the AcpNodeName matches that of the ACP node itself, it would be possible to use the same TA across disjoint ACP domains, but the security and attack implications of such an approach are beyond the scope of this document.

The use of ULA addresses in the AcpNodeName is another novel aspect for certificates from a possible public CA. Typically, ULA addresses are not meant to be signed by a public CA when carried in an address field, because there is no ownership of a particular ULA address in the scope of the Internet, which is what public CA operate on.

Nevertheless, the ULA addresses used by the ACP are scoped to be valid only within the confines of a specific ACP as defined by the domain name in the `AcpNodeName`. However, this understanding has not been reviewed or accepted by any bodies responsible for policies of public CA.

Because in this specification, ACPs are isolated from each other primarily by their TA, when a public CA would intend to sign ACP certificates and using a single TA to sign TA of ACP certificates from different operators/domain, it could do so by ensuring that the domain name in the `AcpNodeName` was a globally owned DNS ACP domain name of the organization, and beyond that, it would need to validate that the ACP registrar of that domain who is mitigating the enrollment is authorized to vouch for the ownership of the `acp-` address within the scope of the ACP domain name.

B.4. Hardening DULL GRASP considerations

DULL GRASP suffers from similar type of DoS attacks as many link-local multicast discovery protocols, for example mDNS. Attackers on a subnet may be able to inject malicious DULL GRASP messages that are indistinguishable from non-malicious DULL GRASP messages to create Denial-of-Service (DoS) attacks that force ACP nodes to attempt many unsuccessful ACP secure channel connections.

When an ACP node sees multiple `AN_ACP` objectives for the same secure channel protocol on different transport addresses, it could prefer connecting via the well-known transport address if the secure channel method has one, such as UDP port 500 for IKEv2. For protocols such as (ACP secure channel over) DTLS for which there are no well defined port number, this heuristic does not provide benefits though.

DoS attack with port numbers can also be eliminated by relying on well known-port numbers implied by the GRASP method-name. For example, a future service name of "DTLSacp" could be defined to be associated only to a newly to be assigned well known UDP port for ACP over DTLS, and the port number in the GRASP transport address information would be ignored. Note that there is already a variety of ports assigned to specific protocols over DTLS by IANA, so especially for DTLS this would not be uncommon.

Authors' Addresses

Toerless Eckert (editor)
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara, 95050
United States of America

Email: tte+ietf@cs.fau.de

Michael H. Behringer (editor)

Email: michael.h.behringer@gmail.com

Steinthor Bjarnason
Arbor Networks
2727 South State Street, Suite 200
Ann Arbor, MI 48104
United States

Email: sbjarnason@arbor.net

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: 15 May 2021

M. Pritikin
Cisco
M. Richardson
Sandelman
T.T.E. Eckert
Futurewei USA
M.H. Behringer

K.W. Watsen
Watsen Networks
11 November 2020

Bootstrapping Remote Secure Key Infrastructures (BRSKI)
draft-ietf-anima-bootstrapping-keyinfra-45

Abstract

This document specifies automated bootstrapping of an Autonomic Control Plane. To do this a Secure Key Infrastructure is bootstrapped. This is done using manufacturer-installed X.509 certificates, in combination with a manufacturer's authorizing service, both online and offline. We call this process the Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol. Bootstrapping a new device can occur using a routable address and a cloud service, or using only link-local connectivity, or on limited/disconnected networks. Support for deployment models with less stringent security requirements is included. Bootstrapping is complete when the cryptographic identity of the new key infrastructure is successfully deployed to the device. The established secure connection can be used to deploy a locally issued certificate to the device as well.

Status of This Memo

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

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

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

This Internet-Draft will expire on 15 May 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction	5
1.1.	Prior Bootstrapping Approaches	6
1.2.	Terminology	8
1.3.	Scope of solution	11
1.3.1.	Support environment	11
1.3.2.	Constrained environments	11
1.3.3.	Network Access Controls	12
1.3.4.	Bootstrapping is not Booting	12
1.4.	Leveraging the new key infrastructure / next steps . . .	12
1.5.	Requirements for Autonomic Network Infrastructure (ANI) devices	13
2.	Architectural Overview	13
2.1.	Behavior of a Pledge	15
2.2.	Secure Imprinting using Vouchers	16
2.3.	Initial Device Identifier	17
2.3.1.	Identification of the Pledge	18
2.3.2.	MASA URI extension	19
2.4.	Protocol Flow	20
2.5.	Architectural Components	23
2.5.1.	Pledge	23
2.5.2.	Join Proxy	23
2.5.3.	Domain Registrar	23
2.5.4.	Manufacturer Service	23
2.5.5.	Public Key Infrastructure (PKI)	24
2.6.	Certificate Time Validation	24
2.6.1.	Lack of realtime clock	24
2.6.2.	Infinite Lifetime of IDevID	24
2.7.	Cloud Registrar	25
2.8.	Determining the MASA to contact	25
3.	Voucher-Request artifact	26

3.1.	Nonceless Voucher Requests	27
3.2.	Tree Diagram	27
3.3.	Examples	27
3.4.	YANG Module	29
4.	Proxying details (Pledge - Proxy - Registrar)	33
4.1.	Pledge discovery of Proxy	34
4.1.1.	Proxy GRASP announcements	35
4.2.	CoAP connection to Registrar	37
4.3.	Proxy discovery and communication of Registrar	37
5.	Protocol Details (Pledge - Registrar - MASA)	38
5.1.	BRSKI-EST TLS establishment details	40
5.2.	Pledge Requests Voucher from the Registrar	41
5.3.	Registrar Authorization of Pledge	43
5.4.	BRSKI-MASA TLS establishment details	43
5.4.1.	MASA authentication of customer Registrar	44
5.5.	Registrar Requests Voucher from MASA	45
5.5.1.	MASA renewal of expired vouchers	47
5.5.2.	MASA pinning of registrar	48
5.5.3.	MASA checking of voucher request signature	48
5.5.4.	MASA verification of domain registrar	49
5.5.5.	MASA verification of pledge prior-signed-voucher-request	50
5.5.6.	MASA nonce handling	50
5.6.	MASA and Registrar Voucher Response	50
5.6.1.	Pledge voucher verification	53
5.6.2.	Pledge authentication of provisional TLS connection	54
5.7.	Pledge BRSKI Status Telemetry	55
5.8.	Registrar audit-log request	56
5.8.1.	MASA audit log response	58
5.8.2.	Calculation of domainID	60
5.8.3.	Registrar audit log verification	61
5.9.	EST Integration for PKI bootstrapping	62
5.9.1.	EST Distribution of CA Certificates	63
5.9.2.	EST CSR Attributes	63
5.9.3.	EST Client Certificate Request	64
5.9.4.	Enrollment Status Telemetry	64
5.9.5.	Multiple certificates	65
5.9.6.	EST over CoAP	66
6.	Clarification of transfer-encoding	66
7.	Reduced security operational modes	66
7.1.	Trust Model	66
7.2.	Pledge security reductions	67
7.3.	Registrar security reductions	68
7.4.	MASA security reductions	69
7.4.1.	Issuing Nonceless vouchers	69
7.4.2.	Trusting Owners on First Use	70
7.4.3.	Updating or extending voucher trust anchors	71

8.	IANA Considerations	72
8.1.	The IETF XML Registry	72
8.2.	YANG Module Names Registry	72
8.3.	BRSKI well-known considerations	72
8.3.1.	BRSKI .well-known registration	72
8.3.2.	BRSKI .well-known registry	73
8.4.	PKIX Registry	73
8.5.	Pledge BRSKI Status Telemetry	73
8.6.	DNS Service Names	74
8.7.	GRASP Objective Names	74
9.	Applicability to the Autonomic Control Plane (ACP)	74
9.1.	Operational Requirements	76
9.1.1.	MASA Operational Requirements	76
9.1.2.	Domain Owner Operational Requirements	77
9.1.3.	Device Operational Requirements	77
10.	Privacy Considerations	78
10.1.	MASA audit log	78
10.2.	What BRSKI-EST reveals	78
10.3.	What BRSKI-MASA reveals to the manufacturer	79
10.4.	Manufacturers and Used or Stolen Equipment	81
10.5.	Manufacturers and Grey market equipment	83
10.6.	Some mitigations for meddling by manufacturers	83
10.7.	Death of a manufacturer	84
11.	Security Considerations	85
11.1.	Denial of Service (DoS) against MASA	86
11.2.	DomainID must be resistant to second-preimage attacks	86
11.3.	Availability of good random numbers	87
11.4.	Freshness in Voucher-Requests	87
11.5.	Trusting manufacturers	88
11.6.	Manufacturer Maintenance of trust anchors	89
11.6.1.	Compromise of Manufacturer IDevID signing keys	91
11.6.2.	Compromise of MASA signing keys	91
11.6.3.	Compromise of MASA web service	93
11.7.	YANG Module Security Considerations	94
12.	Acknowledgements	94
13.	References	94
13.1.	Normative References	94
13.2.	Informative References	98
Appendix A.	IPv4 and non-ANI operations	102
A.1.	IPv4 Link Local addresses	102
A.2.	Use of DHCPv4	102
Appendix B.	mDNS / DNSSD proxy discovery options	102
Appendix C.	Example Vouchers	103
C.1.	Keys involved	103
C.1.1.	Manufacturer Certificate Authority for IDevID signatures	104
C.1.2.	MASA key pair for voucher signatures	105
C.1.3.	Registrar Certificate Authority	107

C.1.4. Registrar key pair	108
C.1.5. Pledge key pair	110
C.2. Example process	111
C.2.1. Pledge to Registrar	111
C.2.2. Registrar to MASA	115
C.2.3. MASA to Registrar	121
Appendix D. Additional References	125
Authors' Addresses	125

1. Introduction

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices that are called pledges in this document. Pledges have an IDevID installed in them at the factory.

"BRSKI" is pronounced like "brewski", a colloquial term for beer in Canada and parts of the US-midwest. [brewski]

This document primarily provides for the needs of the ISP and Enterprise focused ANIMA Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. This bootstrap process satisfies the [RFC7575] requirements of section 3.3 of making all operations secure by default. Other users of the BRSKI protocol will need to provide separate applicability statements that include privacy and security considerations appropriate to that deployment. Section 9 explains the detailed applicability for this the ACP usage.

The BRSKI protocol requires a significant amount of communication between manufacturer and owner: in its default modes it provides a cryptographic transfer of control to the initial owner. In its strongest modes, it leverages sales channel information to identify the owner in advance. Resale of devices is possible, provided that the manufacturer is willing to authorize the transfer. Mechanisms to enable transfers of ownership without manufacturer authorization are not included in this version of the protocol, but could be designed into future versions.

This document describes how pledges discover (or are discovered by) an element of the network domain to which the pledge belongs that will perform the bootstrap. This element (device) is called the registrar. Before any other operation, pledge and registrar need to establish mutual trust:

1. Registrar authenticating the pledge: "Who is this device? What is its identity?"

2. Registrar authorizing the pledge: "Is it mine? Do I want it? What are the chances it has been compromised?"
3. Pledge authenticating the registrar: "What is this registrar's identity?"
4. Pledge authorizing the registrar: "Should I join this network?"

This document details protocols and messages to answer the above questions. It uses a TLS connection and an PKIX-shaped (X.509v3) certificate (an IEEE 802.1AR [IDevID] 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):

- * using the Implicit Trust Anchor [RFC7030] database to authenticate an owner specific service (not an autonomic solution because the URL must be securely distributed),
- * engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- * using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),
- * and using a Certificate-Less TLS mutual authentication method (not an autonomic solution because the distribution of symmetric key material is not autonomic).

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

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

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

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

1.2. Terminology

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

The following terms are defined for clarity:

ANI: The Autonomic Network Infrastructure as defined by [I-D.ietf-anima-reference-model]. Section 9 details specific requirements for pledges, proxies and registrars when they are part of an ANI.

Circuit Proxy: A stateful implementation of the join proxy. This is the assumed type of proxy.

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

Domain: The set of entities that share a common local trust anchor. This includes the proxy, registrar, Domain Certificate Authority, Management components and any existing entity that is already a member of the domain.

domainID: The domain IDentity is a unique value based upon the Registrar CA's certificate. Section 5.8.2 specifies how it is calculated.

Domain CA: The domain Certification Authority (CA) provides certification functionalities to the domain. At a minimum it provides certification functionalities to a registrar and manages the private key that defines the domain. Optionally, it certifies all elements.

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

imprint: The process where a device obtains the cryptographic key material to identify and trust future interactions with a network. This term is taken from Konrad Lorenz's work in biology with new ducklings: during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their

mother. An equivalent for a device is to obtain the fingerprint of the network's root certification authority certificate. A device that imprints on an attacker suffers a similar fate to a duckling that imprints on a hungry wolf. Securely imprinting is a primary focus of this document [imprinting]. The analogy to Lorenz's work was first noted in [Stajano99theresurrecting].

IDevID: An Initial Device Identity X.509 certificate installed by the vendor on new equipment. This is a term from 802.1AR [IDevID]

IPIP Proxy: A stateless proxy alternative.

Join Proxy: A domain entity that helps the pledge join the domain. A join proxy facilitates communication for devices that find themselves in an environment where they are not provided connectivity until after they are validated as members of the domain. For simplicity this document sometimes uses the term of 'proxy' to indicate the join proxy. The pledge is unaware that they are communicating with a proxy rather than directly with a registrar.

Join Registrar (and Coordinator): A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a "join registrar (and coordinator)" to control this process. Typically a join registrar is "inside" its domain. For simplicity this document often refers to this as just "registrar". Within [I-D.ietf-anima-reference-model] this is referred to as the "join registrar autonomic service agent". Other communities use the abbreviation "JRC".

LDevID: A Local Device Identity X.509 certificate installed by the owner of the equipment. This is a term from 802.1AR [LDevID]

manufacturer: the term manufacturer is used throughout this document to be the entity that created the device. This is typically the "original equipment manufacturer" or OEM, but in more complex situations it could be a "value added retailer" (VAR), or possibly even a systems integrator. In general, it a goal of BRSKI to eliminate small distinctions between different sales channels. The reason for this is that it permits a single device, with a uniform firmware load, to be shipped directly to all customers. This eliminates costs for the manufacturer. This also reduces the number of products supported in the field increasing the chance that firmware will be more up to date.

MASA Audit-Log: An anonymized list of previous owners maintained by

the MASA on a per device (per pledge) basis. Described in Section 5.8.1.

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

nonced: a voucher (or request) that contains a nonce (the normal case).

nonceless: a voucher (or request) that does not contain a nonce, relying upon accurate clocks for expiration, or which does not expire.

offline: When an architectural component cannot perform realtime communications with a peer, either due to network connectivity or because the peer is turned off, the operation is said to be occurring offline.

Ownership Tracker: An Ownership Tracker service on the global Internet. The Ownership Tracker uses business processes to accurately track ownership of all devices shipped against domains that have purchased them. Although optional, this component allows vendors to provide additional value in cases where their sales and distribution channels allow for accurate tracking of such ownership. Ownership tracking information is indicated in vouchers as described in [RFC8366]

Pledge: The prospective (unconfigured) device, which has an identity installed at the factory.

(Public) Key Infrastructure: The collection of systems and processes that sustain the activities of a public key system. The registrar acts as an [RFC5280] and [RFC5272] (see section 7) "Registration Authority".

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

Voucher: A signed artifact from the MASA that indicates to a pledge the cryptographic identity of the registrar it should trust. There are different types of vouchers depending on how that trust is asserted. Multiple voucher types are defined in [RFC8366]

1.3. Scope of solution

1.3.1. Support environment

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

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

Nomadic or mobile devices often need to acquire credentials to access the network at the new location. An example of this is mobile phone roaming among network operators, or even between cell towers. This is usually called handoff. BRSKI does not provide a low-latency handoff which is usually a requirement in such situations. For these solutions BRSKI can be used to create a relationship (an LDevID) with the "home" domain owner. The resulting credentials are then used to provide credentials more appropriate for a low-latency handoff.

1.3.2. Constrained environments

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

The solution described in this document is aimed in general at non-constrained (i.e., class 2+ [RFC7228]) devices operating on a non-Challenged network. The entire solution as described here is not intended to be useable as-is by constrained devices operating on challenged networks (such as 802.15.4 Low-power Lossy Networks (LLN)s).

Specifically, there are protocol aspects described here that might result in congestion collapse or energy-exhaustion of intermediate battery powered routers in an LLN. Those types of networks should

not use this solution. These limitations are predominately related to the large credential and key sizes required for device authentication. Defining symmetric key techniques that meet the operational requirements is out-of-scope but the underlying protocol operations (TLS handshake and signing structures) have sufficient algorithm agility to support such techniques when defined.

The imprint protocol described here could, however, be used by non-energy constrained devices joining a non-constrained network (for instance, smart light bulbs are usually mains powered, and speak 802.11). It could also be used by non-constrained devices across a non-energy constrained, but challenged network (such as 802.15.4). The certificate contents, and the process by which the four questions above are resolved do apply to constrained devices. It is simply the actual on-the-wire imprint protocol that could be inappropriate.

1.3.3. Network Access Controls

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

1.3.4. Bootstrapping is not Booting

This document describes "bootstrapping" as the protocol used to obtain a local trust anchor. It is expected that this trust anchor, along with any additional configuration information subsequently installed, is persisted on the device across system restarts ("booting"). Bootstrapping occurs only infrequently such as when a device is transferred to a new owner or has been reset to factory default settings.

1.4. Leveraging the new key infrastructure / next steps

As a result of the protocol described herein, the bootstrapped devices have the Domain CA trust anchor in common. An end entity certificate has optionally been issued from the Domain CA. This makes it possible to securely deploy functionalities across the domain, e.g:

- * Device management.

- * Routing authentication.
- * Service discovery.

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

1.5. Requirements for Autonomic Network Infrastructure (ANI) devices

The BRSKI protocol can be used in a number of environments. Some of the options in this document are the result of requirements that are out of the ANI scope. This section defines the base requirements for ANI devices.

For devices that intend to become part of an Autonomic Network Infrastructure (ANI) ([I-D.ietf-anima-reference-model]) that includes an Autonomic Control Plane ([I-D.ietf-anima-autonomic-control-plane]), the BRSKI protocol MUST be implemented.

The pledge must perform discovery of the proxy as described in Section 4.1 using Generic Autonomic Signaling Protocol (GRASP)'s DULL [I-D.ietf-anima-grasp] M_FLOOD announcements.

Upon successfully validating a voucher artifact, a status telemetry MUST be returned. See Section 5.7.

An ANIMA ANI pledge MUST implement the EST automation extensions described in Section 5.9. They supplement the [RFC7030] EST to better support automated devices that do not have an end user.

The ANI Join Registrar Autonomic Service Agent (ASA) MUST support all the BRSKI and above listed EST operations.

All ANI devices SHOULD support the BRSKI proxy function, using circuit proxies over the ACP. (See Section 4.3)

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components.

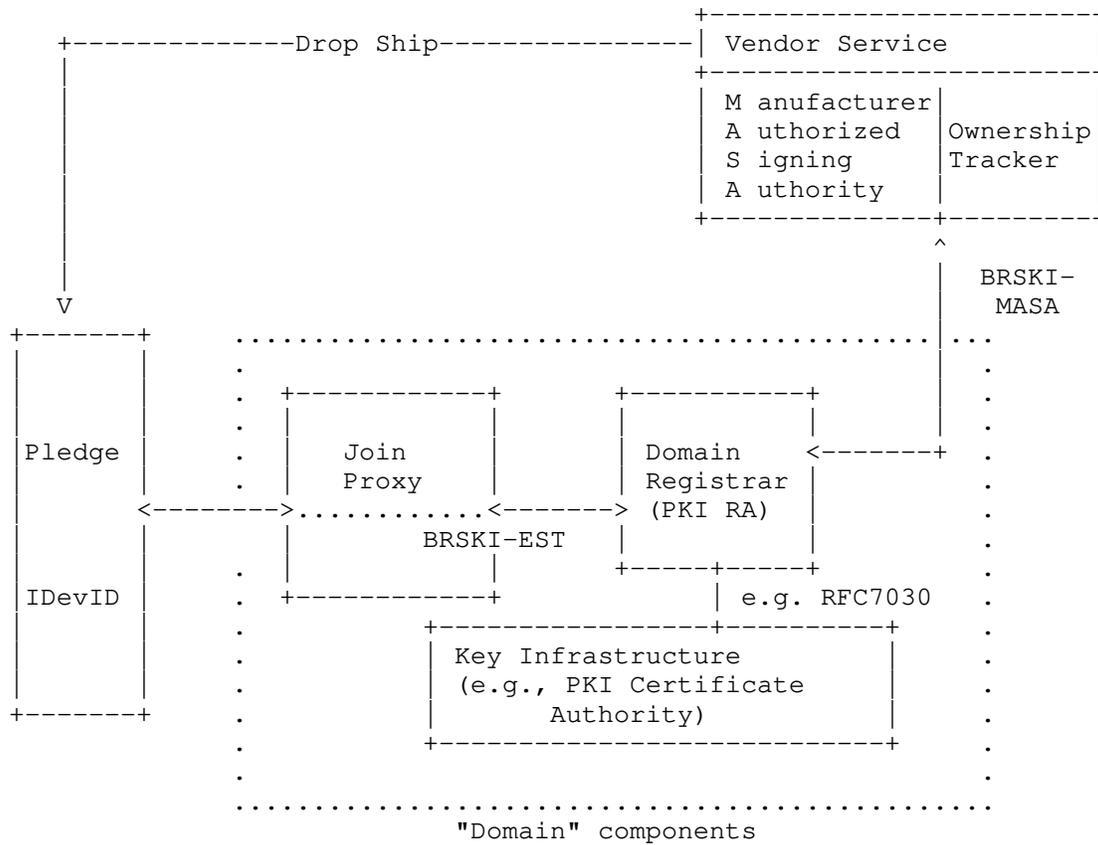


Figure 1: Architecture Overview

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

The domain is the managed network infrastructure with a Key Infrastructure the pledge is joining. The domain provides initial device connectivity sufficient for bootstrapping through a proxy. The domain registrar authenticates the pledge, makes authorization decisions, and distributes vouchers obtained from the Manufacturer Service. Optionally the registrar also acts as a PKI Certification Authority.

2. Identify itself. This is done by presenting an X.509 IDevID credential to the discovered registrar (via the proxy) in a TLS handshake. (The registrar credentials are only provisionally accepted at this time).
3. Request to join the discovered registrar. A unique nonce is included ensuring that any responses can be associated with this particular bootstrapping attempt.
4. Imprint on the registrar. This requires verification of the manufacturer-service-provided voucher. A voucher contains sufficient information for the pledge to complete authentication of a registrar. This document details this step in depth.
5. Enroll. After imprint an authenticated TLS (HTTPS) connection exists between pledge and registrar. Enrollment over Secure Transport (EST) [RFC7030] can then be used to obtain a domain certificate from a registrar.

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

This specification details integration with EST enrollment so that pledges can optionally obtain a locally issued certificate, although any Representational State Transfer (REST) (see [REST]) interface could be integrated in future work.

2.2. Secure Imprinting using Vouchers

A voucher is a cryptographically protected artifact (using a digital signature) to the pledge device authorizing a zero-touch imprint on the registrar domain.

The format and cryptographic mechanism of vouchers is described in detail in [RFC8366].

Vouchers provide a flexible mechanism to secure imprinting: the pledge device only imprints when a voucher can be validated. At the lowest security levels the MASA can indiscriminately issue vouchers and log claims of ownership by domains. At the highest security levels issuance of vouchers can be integrated with complex sales channel integrations that are beyond the scope of this document. The sales channel integration would verify actual (legal) ownership of the pledge by the domain. This provides the flexibility for a number of use cases via a single common protocol mechanism on the pledge and registrar devices that are to be widely deployed in the field. The MASA services have the flexibility to leverage either the currently defined claim mechanisms or to experiment with higher or lower security levels.

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

2.3. Initial Device Identifier

Pledge authentication and pledge voucher-request signing is via a PKIX-shaped certificate installed during the manufacturing process. This is the 802.1AR Initial Device Identifier (IDevID), and it provides a basis for authenticating the pledge during the protocol exchanges described here. There is no requirement for a common root PKI hierarchy. Each device manufacturer can generate its own root certificate. Specifically, the IDevID enables:

1. Uniquely identifying the pledge by the Distinguished Name (DN) and subjectAltName (SAN) parameters in the IDevID. The unique identification of a pledge in the voucher objects are derived from those parameters as described below. Section 10.3 discusses privacy implications of the identifier.
2. Provides a cryptographic authentication of the pledge to the Registrar (see Section 5.3).
3. Secure auto-discovery of the pledge's MASA by the registrar (see Section 2.8).
4. Signing of voucher-request by the pledge's IDevID (see Section 3).
5. Provides a cryptographic authentication of the pledge to the MASA (see Section 5.5.5).

Section 7.2.13 (2009 edition) and section 8.10.3 (2018 edition) of [IDevID] discusses keyUsage and extendedKeyUsage extensions in the IDevID certificate. [IDevID] acknowledges that adding restrictions in the certificate limits applicability of these long-lived certificates. This specification emphasizes this point, and therefore RECOMMENDS that no key usage restrictions be included. This is consistent with [RFC5280] section 4.2.1.3, which does not require key usage restrictions for end entity certificates.

2.3.1. Identification of the Pledge

In the context of BRSKI, pledges have a 1:1 relationship with a "serial-number". This serial-number is used both in the "serial-number" field of voucher or voucher-requests (see Section 3) and in local policies on registrar or MASA (see Section 5).

There is a (certificate) serialNumber field is defined in [RFC5280] section 4.1.2.2. In the ASN.1, this is referred to as the CertificateSerialNumber. This field is NOT relevant to this specification. Do not confuse this field with the "serial-number" defined by this document, or by [IDevID] and [RFC4519] section 2.31.

The device serial number is defined in [RFC5280] section A.1 and A.2 as the X520SerialNumber, with the OID tag id-at-serialNumber.

The device serial number field (X520SerialNumber) is used as follows by the pledge to build the "serial-number" that is placed in the voucher-request. In order to build it, the fields need to be converted into a serial-number of "type string".

An example of a printable form of the "serialNumber" field is provided in [RFC4519] section 2.31 ("WI-3005"). That section further provides equality and syntax attributes.

Due to the reality of existing device identity provisioning processes, some manufacturers have stored serial-numbers in other fields. Registrar's SHOULD be configurable, on a per-manufacturer basis, to look for serial-number equivalents in other fields.

As explained in Section 5.5 the Registrar MUST extract the serial-number again itself from the pledge's TLS certificate. It can consult the serial-number in the pledge-request if there are any possible confusion about the source of the serial-number.

2.3.2. MASA URI extension

This document defines a new PKIX non-critical certificate extension to carry the MASA URI. This extension is intended to be used in the IDevID certificate. The URI is represented as described in Section 7.4 of [RFC5280].

The URI provides the authority information. The BRSKI `"/.well-known"` tree ([RFC5785]) is described in Section 5.

A complete URI MAY be in this extension, including the `'scheme'`, `'authority'`, and `'path'`. The complete URI will typically be used in diagnostic or experimental situations. Typically, (and in consideration to constrained systems), this SHOULD be reduced to only the `'authority'`, in which case a scheme of `"https://"` ([RFC7230] section 2.7.3) and `'path'` of `"/.well-known/brski"` is to be assumed.

The registrar can assume that only the `'authority'` is present in the extension, if there are no slash (`"/"`) characters in the extension.

Section 7.4 of [RFC5280] calls out various schemes that MUST be supported, including LDAP, HTTP and FTP. However, the registrar MUST use HTTPS for the BRSKI-MASA connection.

The new extension is identified as follows:

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

DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS ALL --

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

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

MASACertExtensions EXTENSION ::= { ext-MASAURL, ... }
ext-MASAURL EXTENSION ::= { SYNTAX MASAURLSyntax
IDENTIFIED BY id-pe-masa-url }

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

MASAURLSyntax ::= IA5String

END
<CODE ENDS>
```

Figure 3: MASAURL ASN.1 Module

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

2.4. Protocol Flow

A representative flow is shown in Figure 4

On initial bootstrap, a new device (the pledge) uses a local service autodiscovery (GRASP or mDNS) to locate a join proxy. The join proxy connects the pledge to a local registrar (the JRC).

Having found a candidate registrar, the fledgling pledge sends some information about itself to the registrar, including its serial number in the form of a voucher request and its device identity certificate (IDeVID) as part of the TLS session.

The registrar can determine whether it expected such a device to appear, and locates a MASA. The location of the MASA is usually found in an extension in the IDeVID. Having determined that the MASA is suitable, the entire information from the initial voucher request (including device serial number) is transmitted over the internet in a TLS protected channel to the manufacturer, along with information about the registrar/owner.

The manufacturer can then apply policy based on the provided information, as well as other sources of information (such as sales records), to decide whether to approve the claim by the registrar to own the device; if the claim is accepted, a voucher is issued that directs the device to accept its new owner.

The voucher is returned to the registrar, but not immediately to the device -- the registrar has an opportunity to examine the voucher, the MASA's audit-logs, and other sources of information to determine whether the device has been tampered with, and whether the bootstrap should be accepted.

No filtering of information is possible in the signed voucher, so this is a binary yes-or-no decision. If the registrar accepts the voucher as a proper one for its device, the voucher is returned to the pledge for imprinting.

The voucher also includes a trust anchor that the pledge uses as representing the owner. This is used to successfully bootstrap from an environment where only the manufacturer has built-in trust by the device into an environment where the owner now has a PKI footprint on the device.

When BRSKI is followed with EST this single footprint is further leveraged into the full owner's PKI and a LDeVID for the device. Subsequent reporting steps provide flows of information to indicate success/failure of the process.

2.5. Architectural Components

2.5.1. Pledge

The pledge is the device that is attempting to join. The pledge is assumed to talk to the Join Proxy using link-local network connectivity. In most cases, the pledge has no other connectivity until the pledge completes the enrollment process and receives some kind of network credential.

2.5.2. Join Proxy

The join proxy provides HTTPS connectivity between the pledge and the registrar. A circuit proxy mechanism is described in Section 4. Additional mechanisms, including a CoAP mechanism and a stateless IPIP mechanism are the subject of future work.

2.5.3. Domain Registrar

The domain's registrar operates as the BRSKI-MASA client when requesting vouchers from the MASA (see Section 5.4). The registrar operates as the BRSKI-EST server when pledges request vouchers (see Section 5.1). The registrar operates as the BRSKI-EST server "Registration Authority" if the pledge requests an end entity certificate over the BRSKI-EST connection (see Section 5.9).

The registrar uses an Implicit Trust Anchor database for authenticating the BRSKI-MASA connection's MASA TLS Server Certificate. Configuration or distribution of trust anchors is out-of-scope for this specification.

The registrar uses a different Implicit Trust Anchor database for authenticating the BRSKI-EST connection's Pledge TLS Client Certificate. Configuration or distribution of the BRSKI-EST client trust anchors is out-of-scope of this specification. Note that the trust anchors in/excluded from the database will affect which manufacturers' devices are acceptable to the registrar as pledges, and can also be used to limit the set of MASAs that are trusted for enrollment.

2.5.4. Manufacturer Service

The Manufacturer Service provides two logically separate functions: the Manufacturer Authorized Signing Authority (MASA) described in Section 5.5 and Section 5.6, and an ownership tracking/auditing function described in Section 5.7 and Section 5.8.

2.5.5. Public Key Infrastructure (PKI)

The Public Key Infrastructure (PKI) administers certificates for the domain of concern, providing the trust anchor(s) for it and allowing enrollment of pledges with domain certificates.

The voucher provides a method for the distribution of a single PKI trust anchor (as the "pinned-domain-cert"). A distribution of the full set of current trust anchors is possible using the optional EST integration.

The domain's registrar acts as an [RFC5272] Registration Authority, requesting certificates for pledges from the Key Infrastructure.

The expectations of the PKI are unchanged from EST [RFC7030]. This document does not place any additional architectural requirements on the Public Key Infrastructure.

2.6. Certificate Time Validation

2.6.1. Lack of realtime clock

Many devices when bootstrapping do not have knowledge of the current time. Mechanisms such as Network Time Protocols cannot be secured until bootstrapping is complete. Therefore bootstrapping is defined with a framework that does not require knowledge of the current time. A pledge MAY ignore all time stamps in the voucher and in the certificate validity periods if it does not know the current time.

The pledge is exposed to dates in the following five places: registrar certificate notBefore, registrar certificate notAfter, voucher created-on, and voucher expires-on. Additionally, CMS signatures contain a signingTime.

A pledge with a real time clock in which it has confidence, MUST check the above time fields in all certificates and signatures that it processes.

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

2.6.2. Infinite Lifetime of IDevID

[RFC5280] explains that long lived pledge certificates "SHOULD be assigned the GeneralizedTime value of 99991231235959Z" for the notAfter field.

Some deployed IDevID management systems are not compliant with the 802.1AR requirement for infinite lifetimes, and put in typical ≤ 3 year certificate lifetimes. Registrars SHOULD be configurable on a per-manufacturer basis to ignore pledge lifetimes when the pledge did not follow the RFC5280 recommendations.

2.7. Cloud Registrar

There exist operationally open networks wherein devices gain unauthenticated access to the Internet at large. In these use cases the management domain for the device needs to be discovered within the larger Internet. The case where a device can boot and get access to larger Internet are less likely within the ANIMA ACP scope but may be more important in the future. In the ANIMA ACP scope, new devices will be quarantined behind a Join Proxy.

There are additionally some greenfield situations involving an entirely new installation where a device may have some kind of management uplink that it can use (such as via 3G network for instance). In such a future situation, the device might use this management interface to learn that it should configure itself to become the local registrar.

In order to support these scenarios, the pledge MAY contact a well known URI of a cloud registrar if a local registrar cannot be discovered or if the pledge's target use cases do not include a local registrar.

If the pledge uses a well known URI for contacting a cloud registrar a manufacturer-assigned Implicit Trust Anchor database (see [RFC7030]) MUST be used to authenticate that service as described in [RFC6125]. The use of a DNS-ID for validation is appropriate, and it may include wildcard components on the left-mode side. This is consistent with the human user configuration of an EST server URI in [RFC7030] which also depends on RFC6125.

2.8. Determining the MASA to contact

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

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

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

3. Voucher-Request artifact

Voucher-requests are how vouchers are requested. The semantics of the voucher-request are described below, in the YANG model.

A pledge forms the "pledge voucher-request", signs it with its IDevID and submits it to the registrar.

The registrar in turn forms the "registrar voucher-request", signs it with its Registrar keypair and submits it to the MASA.

The "proximity-registrar-cert" leaf is used in the pledge voucher-requests. This provides a method for the pledge to assert the registrar's proximity.

This network proximity results from the following properties in the ACP context: the pledge is connected to the Join Proxy (Section 4) using a Link-Local IPv6 connection. While the Join Proxy does not participate in any meaningful sense in the cryptography of the TLS connection (such as via a Channel Binding), the Registrar can observe that the connection is via the private ACP (ULA) address of the join proxy, and could not come from outside the ACP. The Pledge must therefore be at most one IPv6 Link-Local hop away from an existing node on the ACP.

Other users of BRSKI will need to define other kinds of assertions if the network proximity described above does not match their needs.

The "prior-signed-voucher-request" leaf is used in registrar voucher-requests. If present, it is the signed pledge voucher-request artifact. This provides a method for the registrar to forward the pledge's signed request to the MASA. This completes transmission of the signed "proximity-registrar-cert" leaf.

Unless otherwise signaled (outside the voucher-request artifact), the signing structure is as defined for vouchers, see [RFC8366].

3.1. Nonceless Voucher Requests

A registrar MAY also retrieve nonceless vouchers by sending nonceless voucher-requests to the MASA in order to obtain vouchers for use when the registrar does not have connectivity to the MASA. No "prior-signed-voucher-request" leaf would be included. The registrar will also need to know the serial number of the pledge. This document does not provide a mechanism for the registrar to learn that in an automated fashion. Typically this will be done via scanning of bar-code or QR-code on packaging, or via some sales channel integration.

3.2. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher-request document. The voucher-request builds upon the voucher artifact described in [RFC8366]. The tree diagram is described in [RFC8340]. Each node in the diagram is fully described by the YANG module in Section 3.4. Please review the YANG module for a detailed description of the voucher-request format.

module: ietf-voucher-request

```

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

```

Figure 5: YANG Tree diagram for Voucher-Request

3.3. Examples

This section provides voucher-request examples for illustration purposes. These examples show the JSON prior to CMS wrapping. JSON encoding rules specify that any binary content be base64 encoded ([RFC4648] section 4). The contents of the (base64) encoded certificates have been elided to save space. For detailed examples, see Appendix C.2. These examples conform to the encoding rules defined in [RFC7951].

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

```
{
  "ietf-voucher-request:voucher": {
    "assertion": "proximity",
    "nonce": "62a2e7693d82fcda2624de58fb6722e5",
    "serial-number" : "JADA123456789",
    "created-on": "2017-01-01T00:00:00.000Z",
    "proximity-registrar-cert": "base64encodedvalue=="
  }
}
```

Figure 6: JSON representation of example Voucher-Request

Example (2) The following example illustrates a registrar voucher-request. The 'prior-signed-voucher-request' leaf is populated with the pledge's voucher-request (such as the prior example). The pledge's voucher-request is a binary CMS signed object. In the JSON encoding used here it must be base64 encoded. The nonce and assertion have been carried forward from the pledge request to the registrar request. The serial-number is extracted from the pledge's Client Certificate from the TLS connection. See Section 5.5.

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

Figure 7: JSON representation of example Prior-Signed Voucher-Request

Example (3) The following example illustrates a registrar voucher-request. The 'prior-signed-voucher-request' leaf is not populated with the pledge's voucher-request nor is the nonce leaf. This form might be used by a registrar requesting a voucher when the pledge can not communicate with the registrar (such as when it is powered down, or

still in packaging), and therefore could not submit a nonce. This scenario is most useful when the registrar is aware that it will not be able to reach the MASA during deployment. See Section 5.5.

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

Figure 8: JSON representation of Offline Voucher-Request

3.4. YANG Module

Following is a YANG [RFC7950] module formally extending the [RFC8366] voucher into a voucher-request.

```
<CODE BEGINS> file "ietf-voucher-request@2018-02-14.yang"
module ietf-voucher-request {
  yang-version 1.1;

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

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

  import ietf-voucher {
    prefix vch;
    description "This module defines the format for a voucher,
      which is produced by a pledge's manufacturer or
      delegate (MASA) to securely assign a pledge to
      an 'owner', so that the pledge may establish a secure
      connection to the owner's network infrastructure";

    reference "RFC 8366: Voucher Artifact for
      Bootstrapping Protocols";
  }

  organization
```

```
"IETF ANIMA Working Group";
```

```
contact
```

```
"WG Web: <https://datatracker.ietf.org/wg/anima/>  
WG List: <mailto:anima@ietf.org>  
Author: Kent Watsen  
<mailto:kent+ietf@watsen.net>  
Author: Michael H. Behringer  
<mailto:Michael.H.Behringer@gmail.com>  
Author: Toerless Eckert  
<mailto:tte+ietf@cs.fau.de>  
Author: Max Pritikin  
<mailto:pritikin@cisco.com>  
Author: Michael Richardson  
<mailto:mcr+ietf@sandelman.ca>";
```

```
description
```

```
"This module defines the format for a voucher request.  
It is a superset of the voucher itself.  
It provides content to the MASA for consideration  
during a voucher request.
```

```
The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL  
NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',  
'MAY', and 'OPTIONAL' in this document are to be interpreted as  
described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,  
they appear in all capitals, as shown here.
```

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

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

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

```
revision "2018-02-14" {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: Bootstrapping Remote Secure Key Infrastructure";  
}
```

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

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

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

    refine "voucher/pinned-domain-cert" {
      mandatory false;
      description "A pinned-domain-cert field
                  is not valid in a voucher request, and
                  any occurrence MUST be ignored";
    }

    refine "voucher/last-renewal-date" {
      description "A last-renewal-date field
                  is not valid in a voucher request, and
                  any occurrence MUST be ignored";
    }

    refine "voucher/domain-cert-revocation-checks" {
      description "The domain-cert-revocation-checks field
                  is not valid in a voucher request, and
                  any occurrence MUST be ignored";
    }

    refine "voucher/assertion" {
      mandatory false;
      description "Any assertion included in registrar voucher
                  requests SHOULD be ignored by the MASA.";
    }

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

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


4. Proxying details (Pledge - Proxy - Registrar)

This section is normative for uses with an ANIMA ACP. The use of the GRASP mechanism is part of the ACP. Other users of BRSKI will need to define an equivalent proxy mechanism, and an equivalent mechanism to configure the proxy.

The role of the proxy is to facilitate communications. The proxy forwards packets between the pledge and a registrar that has been provisioned to the proxy via full GRASP ACP discovery.

This section defines a stateful proxy mechanism which is referred to as a "circuit" proxy. This is a form of Application Level Gateway ([RFC2663] section 2.9).

The proxy does not terminate the TLS handshake: it passes streams of bytes onward without examination. A proxy MUST NOT assume any specific TLS version. Please see [RFC8446] section 9.3 for details on TLS invariants.

A Registrar can directly provide the proxy announcements described below, in which case the announced port can point directly to the Registrar itself. In this scenario the pledge is unaware that there is no proxying occurring. This is useful for Registrars which are servicing pledges on directly connected networks.

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

During the discovery of the Registrar by the Join Proxy, the Join Proxy will also learn which kinds of proxy mechanisms are available. This will allow the Join Proxy to use the lowest impact mechanism which the Join Proxy and Registrar have in common.

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

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

4.1. Pledge discovery of Proxy

The result of discovery is a logical communication with a registrar, through a proxy. The proxy is transparent to the pledge. The communication between the pledge and Join Proxy is over IPv6 Link-Local addresses.

To discover the proxy the pledge performs the following actions:

1. MUST: Obtains a local address using IPv6 methods as described in [RFC4862] IPv6 Stateless Address AutoConfiguration. Use of [RFC4941] temporary addresses is encouraged. To limit pervasive monitoring ([RFC7258]), a new temporary address MAY use a short lifetime (that is, set TEMP_PREFERRED_LIFETIME to be short). Pledges will generally prefer use of IPv6 Link-Local addresses, and discovery of proxy will be by Link-Local mechanisms. IPv4 methods are described in Appendix A
2. MUST: Listen for GRASP M_FLOOD ([I-D.ietf-anima-grasp]) announcements of the objective: "AN_Proxy". See section Section 4.1.1 for the details of the objective. The pledge MAY listen concurrently for other sources of information, see Appendix B.

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

While the GRASP M_FLOOD mechanism is passive for the pledge, the non-normative other methods (mDNS, and IPv4 methods) described in Appendix B are active. The pledge SHOULD run those methods in parallel with listening to for the M_FLOOD. The active methods SHOULD back-off by doubling to a maximum of one hour to avoid overloading the network with discovery attempts. Detection of change of physical link status (Ethernet carrier for instance) SHOULD reset the back off timers.

The pledge could discover more than one proxy on a given physical interface. The pledge can have a multitude of physical interfaces as well: a layer-2/3 Ethernet switch may have hundreds of physical ports.

Each possible proxy offer SHOULD be attempted up to the point where a valid voucher is received: while there are many ways in which the attempt may fail, it does not succeed until the voucher has been validated.

The connection attempts via a single proxy SHOULD exponentially back-off to a maximum of one hour to avoid overloading the network infrastructure. The back-off timer for each MUST be independent of other connection attempts.

Connection attempts SHOULD be run in parallel to avoid head of queue problems wherein an attacker running a fake proxy or registrar could perform protocol actions intentionally slowly. Connection attempts to different proxies SHOULD be sent with an interval of 3 to 5s. The pledge SHOULD continue to listen to for additional GRASP M_FLOOD messages during the connection attempts.

Each connection attempt through a distinct Join Proxy MUST have a unique nonce in the voucher-request.

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

Once all discovered services are attempted (assuming that none succeeded) the device MUST return to listening for GRASP M_FLOOD. It SHOULD periodically retry any manufacturer-specific mechanisms. The pledge MAY prioritize selection order as appropriate for the anticipated environment.

4.1.1. Proxy GRASP announcements

A proxy uses the DULL GRASP M_FLOOD mechanism to announce itself. This announcement can be within the same message as the ACP announcement detailed in [I-D.ietf-anima-autonomic-control-plane].

The formal Concise Data Definition Language (CDDL) [RFC8610] definition is:

```

<CODE BEGINS> file "proxygrasp.cddl"
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]

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

ttl          = 180000          ; 180,000 ms (3 minutes)
initiator    = ACP address to contact Registrar
objective-flags = sync-only   ; as in GRASP spec
sync-only    = 4              ; M_FLOOD only requires synchronization
loop-count   = 1              ; one hop only
objective-value = any         ; none

locator-option = [ O_IPv6_LOCATOR, ipv6-address,
                  transport-proto, port-number ]
ipv6-address   = the v6 LL of the Proxy
$transport-proto /= IPPROTO_TCP ; note this can be any value from the
                                ; IANA protocol registry, as per
                                ; [GRASP] section 2.9.5.1, note 3.
port-number    = selected by Proxy
<CODE ENDS>

```

Figure 10: CDDL definition of Proxy Discovery message

Here is an example M_FLOOD announcing a proxy at fe80::1, on TCP port 4443.

```

[M_FLOOD, 12340815, h'fe800000000000000000000000000001', 180000,
  ["AN_Proxy", 4, 1, ""],
  [O_IPv6_LOCATOR,
   h'fe800000000000000000000000000001', IPPROTO_TCP, 4443]]

```

Figure 11: Example of Proxy Discovery message

On a small network the Registrar MAY include the GRASP M_FLOOD announcements to locally connected networks.

The \$transport-proto above indicates the method that the pledge-proxy-registrar will use. The TCP method described here is mandatory, and other proxy methods, such as CoAP methods not defined in this document are optional. Other methods MUST NOT be enabled unless the Join Registrar ASA indicates support for them in its own announcement.

4.2. CoAP connection to Registrar

The use of CoAP to connect from pledge to registrar is out of scope for this document, and is described in future work. See [I-D.ietf-anima-constrained-voucher].

4.3. Proxy discovery and communication of Registrar

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

The registrar announces itself using ACP instance of GRASP using M_FLOOD messages, with the "AN_join_registrar" objective. A registrar may announce any convenient port number, including using a stock port 443. ANI proxies MUST support GRASP discovery of registrars.

The M_FLOOD is formatted as follows:

```
[M_FLOOD, 51804321, h'fda379a6f6ee00000200000064000001', 180000,
  [{"AN_join_registrar", 4, 255, "EST-TLS"},
  [O_IPv6_LOCATOR,
   h'fda379a6f6ee00000200000064000001', IPPROTO_TCP, 8443]]]
```

Figure 12: An example of a Registrar announcement message

The formal CDDL definition is:

```
<CODE BEGINS> file "jrcgrasp.cddl"
flood-message = [M_FLOOD, session-id, initiator, ttl,
  +[objective, (locator-option / [])]]

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

initiator = ACP address to contact Registrar
objective-flags = sync-only ; as in GRASP spec
sync-only = 4 ; M_FLOOD only requires synchronization
loop-count = 255 ; mandatory maximum
objective-value = text ; name of the (list of) of supported
; protocols: "EST-TLS" for RFC7030.

<CODE ENDS>
```

Figure 13: CDDL definition for Registrar announcement message

The M_FLOOD message MUST be sent periodically. The default period SHOULD be 60 seconds, the value SHOULD be operator configurable but SHOULD NOT be smaller than 60 seconds. The frequency of sending MUST be such that the aggregate amount of periodic M_FLOODs from all flooding sources cause only negligible traffic across the ACP.

Here are some examples of locators for illustrative purposes. Only the first one (\$transport-protocol = 6, TCP) is defined in this document and is mandatory to implement.

```
locator1 = [O_IPv6_LOCATOR, fd45:1345::6789, 6, 443]
locator2 = [O_IPv6_LOCATOR, fd45:1345::6789, 17, 5683]
locator3 = [O_IPv6_LOCATOR, fe80::1234, 41, nil]
```

A protocol of 6 indicates that TCP proxying on the indicated port is desired.

Registrars MUST announce the set of protocols that they support. They MUST support TCP traffic.

Registrars MUST accept HTTPS/EST traffic on the TCP ports indicated.

Registrars MUST support ANI TLS circuit proxy and therefore BRSKI across HTTPS/TLS native across the ACP.

In the ANI, the Autonomic Control Plane (ACP) secured instance of GRASP ([I-D.ietf-anima-grasp]) MUST be used for discovery of ANI registrar ACP addresses and ports by ANI proxies. The TCP leg of the proxy connection between ANI proxy and ANI registrar therefore also runs across the ACP.

5. Protocol Details (Pledge - Registrar - MASA)

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

BRSKI is described as extensions to EST [RFC7030]. The goal of these extensions is to reduce the number of TLS connections and crypto operations required on the pledge. The registrar implements the BRSKI REST interface within the `"/.well-known/brski"` URI tree, as well as implementing the existing EST URIs as described in EST [RFC7030] section 3.2.2. The communication channel between the pledge and the registrar is referred to as "BRSKI-EST" (see Figure 1).

The communication channel between the registrar and MASA is a new communication channel, similar to EST, within the newly registered `"/.well-known/brski"` tree. For clarity this channel is referred to as `"BRSKI-MASA"`. (See Figure 1).

The MASA URI is `"https://"` authority `"/.well-known/brski"`.

BRSKI uses existing CMS message formats for existing EST operations. BRSKI uses JSON [RFC8259] for all new operations defined here, and voucher formats. In all places where a binary value must be carried in a JSON string, the use of base64 format ([RFC4648] section 4) is to be used, as per [RFC7951] section 6.6.

While EST section 3.2 does not insist upon use of HTTP persistent connections ([RFC7230] section 6.3), BRSKI-EST connections SHOULD use persistent connections. The intention of this guidance is to ensure the provisional TLS state occurs only once, and that the subsequent resolution of the provision state is not subject to a MITM attack during a critical phase.

If non-persistent connections are used, then both the pledge and the registrar MUST remember the certificates seen, and also sent for the first connection. They MUST check each subsequent connections for the same certificates, and each end MUST use the same certificates as well. This places a difficult restriction on rolling certificates on the Registrar.

Summarized automation extensions for the BRSKI-EST flow are:

- * The pledge either attempts concurrent connections via each discovered proxy, or it times out quickly and tries connections in series, as explained at the end of Section 5.1.
- * The pledge provisionally accepts the registrar certificate during the TLS handshake as detailed in Section 5.1.
- * The pledge requests a voucher using the new REST calls described below. This voucher is then validated.
- * The pledge completes authentication of the server certificate as detailed in Section 5.6.1. This moves the BRSKI-EST TLS connection out of the provisional state.
- * Mandatory bootstrap steps conclude with voucher status telemetry (see Section 5.7).

The BRSKI-EST TLS connection can now be used for EST enrollment.

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

- * Client authentication is automated using Initial Device Identity (IDevID) as per the EST certificate based client authentication. The subject field's DN encoding MUST include the "serialNumber" attribute with the device's unique serial number as explained in Section 2.3.1
- * The registrar requests and validates the voucher from the MASA.
- * The registrar forwards the voucher to the pledge when requested.
- * The registrar performs log verifications (described in Section 5.8.3) in addition to local authorization checks before accepting optional pledge device enrollment requests.

5.1. BRSKI-EST TLS establishment details

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

Use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED on the Pledge side. TLS 1.3 (or newer) SHOULD be available on the Registrar server interface, and the Registrar client interface, but TLS 1.2 MAY be used. TLS 1.3 (or newer) SHOULD be available on the MASA server interface, but TLS 1.2 MAY be used.

Establishment of the BRSKI-EST TLS connection is as specified in EST [RFC7030] section 4.1.1 "Bootstrap Distribution of CA Certificates" [RFC7030] wherein the client is authenticated with the IDevID certificate, and the EST server (the registrar) is provisionally authenticated with an unverified server certificate. Configuration or distribution of the trust anchor database used for validating the IDevID certificate is out-of-scope of this specification. Note that the trust anchors in/excluded from the database will affect which manufacturers' devices are acceptable to the registrar as pledges, and can also be used to limit the set of MASAs that are trusted for enrollment.

The signature in the certificate MUST be validated even if a signing key can not (yet) be validated. The certificate (or chain) MUST be retained for later validation.

A self-signed certificate for the Registrar is acceptable as the voucher can validate it upon successful enrollment.

The pledge performs input validation of all data received until a voucher is verified as specified in Section 5.6.1 and the TLS connection leaves the provisional state. Until these operations are complete the pledge could be communicating with an attacker.

The pledge code needs to be written with the assumption that all data is being transmitted at this point to an unauthenticated peer, and that received data, while inside a TLS connection, **MUST** be considered untrusted. This particularly applies to HTTP headers and CMS structures that make up the voucher.

A pledge that can connect to multiple Registrars concurrently **SHOULD** do so. Some devices may be unable to do so for lack of threading, or resource issues. Concurrent connections defeat attempts by a malicious proxy from causing a TCP Slowloris-like attack (see [slowloris]).

A pledge that can not maintain as many connections as there are eligible proxies will need to rotate among the various choices, terminating connections that do not appear to be making progress. If no connection is making progress after 5 seconds then the pledge **SHOULD** drop the oldest connection and go on to a different proxy: the proxy that has been communicated with least recently. If there were no other proxies discovered, the pledge **MAY** continue to wait, as long as it is concurrently listening for new proxy announcements.

5.2. Pledge Requests Voucher from the Registrar

When the pledge bootstraps it makes a request for a voucher from a registrar.

This is done with an HTTPS POST using the operation path value of `"/.well-known/brski/requestvoucher"`.

The pledge voucher-request Content-Type is:

`application/voucher-cms+json` [RFC8366] defines a "YANG-defined JSON document that has been signed using a CMS structure", and the voucher-request described in Section 3 is created in the same way. The media type is the same as defined in [RFC8366]. This is also used for the pledge voucher-request. The pledge **MUST** sign the request using the Section 2.3 credential.

Registrar implementations **SHOULD** anticipate future media types but of course will simply fail the request if those types are not yet known.

The pledge SHOULD include an [RFC7231] section 5.3.2 "Accept" header field indicating the acceptable media type for the voucher response. The "application/voucher-cms+json" media type is defined in [RFC8366] but constrained voucher formats are expected in the future. Registrars and MASA are expected to be flexible in what they accept.

The pledge populates the voucher-request fields as follows:

created-on: Pledges that have a realtime clock are RECOMMENDED to populate this field with the current date and time in yang:date-and-time format. This provides additional information to the MASA. Pledges that have no real-time clocks MAY omit this field.

nonce: The pledge voucher-request MUST contain a cryptographically strong random or pseudo-random number nonce (see [RFC4086] section 6.2). As the nonce is usually generated very early in the boot sequence there is a concern that the same nonce might be generated across multiple boots, or after a factory reset. Different nonces MUST be generated for each bootstrapping attempt, whether in series or concurrently. The freshness of this nonce mitigates against the lack of real-time clock as explained in Section 2.6.1.

assertion: The pledge indicates support for the mechanism described in this document, by putting the value "proximity" in the voucher-request, MUST include the "proximity-registrar-cert" field (below).

proximity-registrar-cert: In a pledge voucher-request this is the first certificate in the TLS server 'certificate_list' sequence (see [RFC5246]) presented by the registrar to the pledge. That is, it is the end-entity certificate. This MUST be populated in a pledge voucher-request.

serial-number The serial number of the pledge is included in the voucher-request from the Pledge. This value is included as a sanity check only, but it is not to be forwarded by the Registrar as described in Section 5.5.

All other fields MAY be omitted in the pledge voucher-request.

An example JSON payload of a pledge voucher-request is in Section 3.3 Example 1.

The registrar confirms that the assertion is 'proximity' and that pinned 'proximity-registrar-cert' is the Registrar's certificate. If this validation fails, then there is an On-Path Attacker (MITM), and the connection MUST be closed after the returning an HTTP 401 error code.

5.3. Registrar Authorization of Pledge

In a fully automated network all devices must be securely identified and authorized to join the domain.

A Registrar accepts or declines a request to join the domain, based on the authenticated identity presented. For different networks, examples of automated acceptance may include:

- * allow any device of a specific type (as determined by the X.509 IDevID),
- * allow any device from a specific vendor (as determined by the X.509 IDevID),
- * allow a specific device from a vendor (as determined by the X.509 IDevID) against a domain white list. (The mechanism for checking a shared white list potentially used by multiple Registrars is out of scope).

If validation fails the registrar SHOULD respond with the HTTP 404 error code. If the voucher-request is in an unknown format, then an HTTP 406 error code is more appropriate. A situation that could be resolved with administrative action (such as adding a vendor to a whitelist) MAY be responded with an 403 HTTP error code.

If authorization is successful the registrar obtains a voucher from the MASA service (see Section 5.5) and returns that MASA signed voucher to the pledge as described in Section 5.6.

5.4. BRSKI-MASA TLS establishment details

The BRSKI-MASA TLS connection is a 'normal' TLS connection appropriate for HTTPS REST interfaces. The registrar initiates the connection and uses the MASA URL obtained as described in Section 2.8. The mechanisms in [RFC6125] SHOULD be used in authentication of the MASA using a DNS-ID that matches that which is found in the IDevID. Registrars MAY include a mechanism to override the MASA URL on a manufacturer-by-manufacturer basis, and within that override it is appropriate to provide alternate anchors. This will typically be used by some vendors to establish explicit (or private) trust anchors for validating their MASA that is part of a sales channel integration.

Use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED. TLS 1.3 (or newer) SHOULD be available.

As described in [RFC7030], the MASA and the registrars SHOULD be prepared to support TLS client certificate authentication and/or HTTP Basic, Digest, or SCRAM authentication. This connection MAY also have no client authentication at all.

Registrars SHOULD permit trust anchors to be pre-configured on a per-vendor(MASA) basis. Registrars SHOULD include the ability to configure a TLS ClientCertificate on a per-MASA basis, or to use no client certificate. Registrars SHOULD also permit HTTP Basic and Digest authentication to be configured.

The authentication of the BRSKI-MASA connection does not change the voucher-request process, as voucher-requests are already signed by the registrar. Instead, this authentication provides access control to the audit-log as described in Section 5.8.

Implementors are advised that contacting the MASA is to establish a secured API connection with a web service and that there are a number of authentication models being explored within the industry. Registrars are RECOMMENDED to fail gracefully and generate useful administrative notifications or logs in the advent of unexpected HTTP 401 (Unauthorized) responses from the MASA.

5.4.1. MASA authentication of customer Registrar

Providing per-customer options requires that the customer's registrar be uniquely identified. This can be done by any stateless method that HTTPS supports such as with HTTP Basic or Digest authentication (that is using a password), but the use of TLS Client Certificate authentication is RECOMMENDED.

Stateful methods involving API tokens, or HTTP Cookies, are not recommended.

It is expected that the setup and configuration of per-customer Client Certificates is done as part of a sales ordering process.

The use of public PKI (i.e. WebPKI) End-Entity Certificates to identify the Registrar is reasonable, and if done universally this would permit a MASA to identify a customers' Registrar simply by a FQDN.

The use of DANE records in DNSSEC signed zones would also permit use of a FQDN to identify customer Registrars.

A third (and simplest, but least flexible) mechanism would be for the MASA to simply store the Registrar's certificate pinned in a database.

A MASA without any supply chain integration can simply accept Registrars without any authentication, or can accept them on a blind Trust-on-First-Use basis as described in Section 7.4.2.

This document does not make a specific recommendation on how the MASA authenticates the Registrar as there are likely different tradeoffs in different environments and product values. Even within the ANIMA ACP applicability, there is a significant difference between supply chain logistics for \$100 CPE devices and \$100,000 core routers.

5.5. Registrar Requests Voucher from MASA

When a registrar receives a pledge voucher-request it in turn submits a registrar voucher-request to the MASA service via an HTTPS interface ([RFC7231]).

This is done with an HTTP POST using the operation path value of `"/.well-known/brski/requestvoucher"`.

The voucher media type `"application/voucher-cms+json"` is defined in [RFC8366] and is also used for the registrar voucher-request. It is a JSON document that has been signed using a CMS structure. The registrar MUST sign the registrar voucher-request.

MASA implementations SHOULD anticipate future media ntypes but of course will simply fail the request if those types are not yet known.

The voucher-request CMS object includes some number of certificates that are input to the MASA as it populates the `'pinned-domain-cert'`. As the [RFC8366] is quite flexible in what may be put into the `'pinned-domain-cert'`, the MASA needs some signal as to what certificate would be effective to populate the field with: it may range from the End Entity (EE) Certificate that the Registrar uses, to the entire private Enterprise CA certificate. More specific certificates result in a tighter binding of the voucher to the domain, while less specific certificates result in more flexibility in how the domain is represented by certificates.

A Registrar which is seeking a nonceless voucher for later offline use benefits from a less specific certificate, as it permits the actual keypair used by a future Registrar to be determined by the pinned certificate authority.

In some cases, a less specific certificate, such a public WebPKI certificate authority, could be too open, and could permit any entity issued a certificate by that authority to assume ownership of a device that has a voucher pinned. Future work may provide a solution to pin both a certificate and a name that would reduce such risk of malicious ownership assertions.

The Registrar SHOULD request a voucher with the most specificity consistent with the mode that it is operating in. In order to do this, when the Registrar prepares the CMS structure for the signed voucher-request, it SHOULD include only certificates which are part of the chain that it wishes the MASA to pin. This MAY be as small as only the End-Entity certificate (with id-kp-cmcRA set) that it uses as it's TLS Server Certificate, or it MAY be the entire chain, including the Domain CA.

The Registrar SHOULD include an [RFC7231] section 5.3.2 "Accept" header field indicating the response media types that are acceptable. This list SHOULD be the entire list presented to the Registrar in the Pledge's original request (see Section 5.2) but MAY be a subset. The MASA is expected to be flexible in what it accepts.

The registrar populates the voucher-request fields as follows:

created-on: The Registrars SHOULD populate this field with the current date and time when the Registrar formed this voucher request. This field provides additional information to the MASA.

nonce: This value, if present, is copied from the pledge voucher-request. The registrar voucher-request MAY omit the nonce as per Section 3.1.

serial-number: The serial number of the pledge the registrar would like a voucher for. The registrar determines this value by parsing the authenticated pledge IDevID certificate. See Section 2.3. The registrar MUST verify that the serial number field it parsed matches the serial number field the pledge provided in its voucher-request. This provides a sanity check useful for detecting error conditions and logging. The registrar MUST NOT simply copy the serial number field from a pledge voucher request as that field is claimed but not certified.

idevid-issuer: The Issuer value from the pledge IDevID certificate is included to ensure unique interpretation of the serial-number. In the case of nonceless (offline) voucher-request, then an appropriate value needs to be configured from the same out-of-band source as the serial-number.

prior-signed-voucher-request: The signed pledge voucher-request SHOULD be included in the registrar voucher-request. The entire CMS signed structure is to be included, base64 encoded for transport in the JSON structure.

A nonceless registrar voucher-request MAY be submitted to the MASA. Doing so allows the registrar to request a voucher when the pledge is offline, or when the registrar anticipates not being able to connect to the MASA while the pledge is being deployed. Some use cases require the registrar to learn the appropriate IDevID SerialNumber field and appropriate 'Accept header field' values from the physical device labeling or from the sales channel (out-of-scope for this document).

All other fields MAY be omitted in the registrar voucher-request.

The "proximity-registrar-cert" field MUST NOT be present in the registrar voucher-request.

Example JSON payloads of registrar voucher-requests are in Section 3.3 Examples 2 through 4.

The MASA verifies that the registrar voucher-request is internally consistent but does not necessarily authenticate the registrar certificate since the registrar MAY be unknown to the MASA in advance. The MASA performs the actions and validation checks described in the following sub-sections before issuing a voucher.

5.5.1. MASA renewal of expired vouchers

As described in [RFC8366] vouchers are normally short lived to avoid revocation issues. If the request is for a previous (expired) voucher using the same registrar (that is, a Registrar with the same Domain CA) then the request for a renewed voucher SHOULD be automatically authorized. The MASA has sufficient information to determine this by examining the request, the registrar authentication, and the existing audit-log. The issuance of a renewed voucher is logged as detailed in Section 5.6.

To inform the MASA that existing vouchers are not to be renewed one can update or revoke the registrar credentials used to authorize the request (see Section 5.5.4 and Section 5.5.3). More flexible methods will likely involve sales channel integration and authorizations (details are out-of-scope of this document).

5.5.2. MASA pinning of registrar

A certificate chain is extracted from the Registrar's signed CMS container. This chain may be as short as a single End-Entity Certificate, up to the entire registrar certificate chain, including the Domain CA certificate, as specified in Section 5.5.

If the domain's CA is unknown to the MASA, then it is to be considered a temporary trust anchor for the rest of the steps in this section. The intention is not to authenticate the message as having come from a fully validated origin, but to establish the consistency of the domain PKI.

The MASA MAY use the certificate farthest in the chain chain that it received from the Registrar from the end-entity, as determined by MASA policy. A MASA MAY have a local policy that it only pins the End-Entity certificate. This is consistent with [RFC8366]. Details of the policy will typically depend upon the degree of Supply Chain Integration, and the mechanism used by the Registrar to authenticate. Such a policy would also determine how the MASA will respond to a request for a nonceless voucher.

5.5.3. MASA checking of voucher request signature

As described in Section 5.5.2, the MASA has extracted Registrar's domain CA. This is used to validate the CMS signature ([RFC5652]) on the voucher-request.

Normal PKIX revocation checking is assumed during voucher-request signature validation. This CA certificate MAY have Certificate Revocation List distribution points, or Online Certificate Status Protocol (OCSP) information ([RFC6960]). If they are present, the MASA MUST be able to reach the relevant servers belonging to the Registrar's domain CA to perform the revocation checks.

The use of OCSP Stapling is preferred.

5.5.4. MASA verification of domain registrar

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

Even when a domain CA is authenticated to the MASA, and there is strong sales channel integration to understand who the legitimate owner is, the above id-kp-cmcRA check prevents arbitrary End-Entity certificates (such as an LDevID certificate) from having vouchers issued against them.

Other cases of inappropriate voucher issuance are detected by examination of the audit log.

If a nonceless voucher-request is submitted the MASA MUST authenticate the registrar as described in either EST [RFC7030] section 3.2.3, section 3.3.2, or by validating the registrar's certificate used to sign the registrar voucher-request using a configured trust anchor. Any of these methods reduce the risk of DDoS attacks and provide an authenticated identity as an input to sales channel integration and authorizations (details are out-of-scope of this document).

In the nonced case, validation of the Registrar's identity (via TLS Client Certificate or HTTP authentication) MAY be omitted if the device policy is to accept audit-only vouchers.

5.5.5. MASA verification of pledge prior-signed-voucher-request

The MASA MAY verify that the registrar voucher-request includes the 'prior-signed-voucher-request' field. If so the prior-signed-voucher-request MUST include a 'proximity-registrar-cert' that is consistent with the certificate used to sign the registrar voucher-request. Additionally the voucher-request serial-number leaf MUST match the pledge serial-number that the MASA extracts from the signing certificate of the prior-signed-voucher-request. The consistency check described above is checking that the 'proximity-registrar-cert' SPKI fingerprint exists within the registrar voucher-request CMS signature's certificate chain. This is substantially the same as the pin validation described in in [RFC7469] section 2.6, paragraph three.

If these checks succeed the MASA updates the voucher and audit-log assertion leafs with the "proximity" assertion, as defined by [RFC8366] section 5.3.

5.5.6. MASA nonce handling

The MASA does not verify the nonce itself. If the registrar voucher-request contains a nonce, and the prior-signed-voucher-request exists, then the MASA MUST verify that the nonce is consistent. (Recall from above that the voucher-request might not contain a nonce, see Section 5.5 and Section 5.5.4).

The MASA populates the audit-log with the nonce that was verified. If a nonceless voucher is issued, then the audit-log is to be populated with the JSON value "null".

5.6. MASA and Registrar Voucher Response

The MASA voucher response to the registrar is forwarded without changes to the pledge; therefore this section applies to both the MASA and the registrar. The HTTP signaling described applies to both the MASA and registrar responses.

When a voucher request arrives at the registrar, if it has a cached response from the MASA for the corresponding registrar voucher-request, that cached response can be used according to local policy; otherwise the registrar constructs a new registrar voucher-request and sends it to the MASA.

Registrar evaluation of the voucher itself is purely for transparency and audit purposes to further inform log verification (see Section 5.8.3) and therefore a registrar could accept future voucher formats that are opaque to the registrar.

If the voucher-request is successful, the server (MASA responding to registrar or registrar responding to pledge) response MUST contain an HTTP 200 response code. The server MUST answer with a suitable 4xx or 5xx HTTP [RFC7230] error code when a problem occurs. In this case, the response data from the MASA MUST be a plaintext human-readable (UTF-8) error message containing explanatory information describing why the request was rejected.

The registrar MAY respond with an HTTP 202 ("the request has been accepted for processing, but the processing has not been completed") as described in EST [RFC7030] section 4.2.3 wherein the client "MUST wait at least the specified 'Retry-After' time before repeating the same request". (see [RFC7231] section 6.6.4) The pledge is RECOMMENDED to provide local feedback (blinking LED etc) during this wait cycle if mechanisms for this are available. To prevent an attacker registrar from significantly delaying bootstrapping the pledge MUST limit the 'Retry-After' time to 60 seconds. Ideally the pledge would keep track of the appropriate Retry-After header field values for any number of outstanding registrars but this would involve a state table on the pledge. Instead the pledge MAY ignore the exact Retry-After value in favor of a single hard coded value (a registrar that is unable to complete the transaction after the first 60 seconds has another chance a minute later). A pledge SHOULD only maintain a 202 retry-state for up to 4 days, which is longer than a long weekend, after which time the enrollment attempt fails and the pledge returns to discovery state.

A pledge that retries a request after receiving a 202 message MUST resend the same voucher-request. It MUST NOT sign a new voucher-request each time, and in particular, it MUST NOT change the nonce value.

In order to avoid infinite redirect loops, which a malicious registrar might do in order to keep the pledge from discovering the correct registrar, the pledge MUST NOT follow more than one redirection (3xx code) to another web origin. EST supports redirection but requires user input; this change allows the pledge to follow a single redirection without a user interaction.

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

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

A 406 (Not Acceptable) response is appropriate if a voucher of the desired type or using the desired algorithms (as indicated by the `Accept:` header fields, and algorithms used in the signature) cannot be issued such as because the MASA knows the pledge cannot process that type. The registrar SHOULD use this response if it determines the pledge is unacceptable due to inventory control, MASA audit-logs, or any other reason.

A 415 (Unsupported Media Type) response is appropriate for a request that has a `voucher-request` or `Accept:` value that is not understood.

The voucher response format is as indicated in the submitted `Accept` header fields or based on the MASA's prior understanding of proper format for this Pledge. Only the [RFC8366] `"application/voucher-cms+json"` media type is defined at this time. The syntactic details of vouchers are described in detail in [RFC8366]. Figure 14 shows a sample of the contents of a voucher.

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

Figure 14: An example voucher

The MASA populates the voucher fields as follows:

`nonce`: The nonce from the pledge if available. See Section 5.5.6.

`assertion`: The method used to verify the relationship between pledge and registrar. See Section 5.5.5.

`pinned-domain-cert`: A certificate. See Section 5.5.2. This figure is illustrative, for an example, see Appendix C.2 where an End Entity certificate is used.

`serial-number`: The serial-number as provided in the `voucher-request`. Also see Section 5.5.5.

`domain-cert-revocation-checks`: Set as appropriate for the pledge's

capabilities and as documented in [RFC8366]. The MASA MAY set this field to 'false' since setting it to 'true' would require that revocation information be available to the pledge and this document does not make normative requirements for [RFC6961] or equivalent integrations.

expires-on: This is set for nonceless vouchers. The MASA ensures the voucher lifetime is consistent with any revocation or pinned-domain-cert consistency checks the pledge might perform. See section Section 2.6.1. There are three times to consider: (a) a configured voucher lifetime in the MASA, (b) the expiry time for the registrar's certificate, (c) any certificate revocation information (CRL) lifetime. The expires-on field SHOULD be before the earliest of these three values. Typically (b) will be some significant time in the future, but (c) will typically be short (on the order of a week or less). The RECOMMENDED period for (a) is on the order of 20 minutes, so it will typically determine the lifespan of the resulting voucher. 20 minutes is sufficient time to reach the post-provisional state in the pledge, at which point there is an established trust relationship between pledge and registrar. The subsequent operations can take as long as required from that point onwards. The lifetime of the voucher has no impact on the lifespan of the ownership relationship.

Whenever a voucher is issued the MASA MUST update the audit-log sufficiently to generate the response as described in Section 5.8.1. The internal state requirements to maintain the audit-log are out-of-scope.

5.6.1. Pledge voucher verification

The pledge MUST verify the voucher signature using the manufacturer-installed trust anchor(s) associated with the manufacturer's MASA (this is likely included in the pledge's firmware). Management of the manufacturer-installed trust anchor(s) is out-of-scope of this document; this protocol does not update these trust anchor(s).

The pledge MUST verify the serial-number field of the signed voucher matches the pledge's own serial-number.

The pledge MUST verify the nonce information in the voucher. If present, the nonce in the voucher must match the nonce the pledge submitted to the registrar; vouchers with no nonce can also be accepted (according to local policy, see Section 7.2)

The pledge MUST be prepared to parse and fail gracefully from a voucher response that does not contain a 'pinned-domain-cert' field. Such a thing indicates a failure to enroll in this domain, and the pledge MUST attempt joining with other available Join Proxy.

The pledge MUST be prepared to ignore additional fields that it does not recognize.

5.6.2. Pledge authentication of provisional TLS connection

Following the process described in [RFC8366], the pledge should consider the public key from the pinned-domain-cert as the sole temporary trust anchor.

The pledge then evaluates the TLS Server Certificate chain that it received when the TLS connection was formed using this trust anchor. It is possible that the pinned-domain-cert matches the End-Entity Certificate provided in the TLS Server.

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

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

The pledge's PKIX path validation of a registrar certificate's validity period information is as described in Section 2.6.1. Once the PKIX path validation is successful the TLS connection is no longer provisional.

The pinned-domain-cert MAY be installed as a trust anchor for future operations such as enrollment (e.g. [RFC7030] as recommended) or trust anchor management or raw protocols that do not need full PKI based key management. It can be used to authenticate any dynamically discovered EST server that contain the id-kp-cmcRA extended key usage extension as detailed in EST RFC7030 section 3.6.1; but to reduce system complexity the pledge SHOULD avoid additional discovery

operations. Instead the pledge SHOULD communicate directly with the registrar as the EST server. The 'pinned-domain-cert' is not a complete distribution of the [RFC7030] section 4.1.3 CA Certificate Response, which is an additional justification for the recommendation to proceed with EST key management operations. Once a full CA Certificate Response is obtained it is more authoritative for the domain than the limited 'pinned-domain-cert' response.

5.7. Pledge BRSKI Status Telemetry

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

The pledge MUST indicate its pledge status regarding the voucher. It does this by sending a status message to the Registrar.

The posted data media type: application/json

The client sends an HTTP POST to the server at the URI ".well-known/brski/voucher_status".

The format and semantics described below are for version 1. A version field is included to permit significant changes to this feedback in the future. A Registrar that receives a status message with a version larger than it knows about SHOULD log the contents and alert a human.

The Status field indicates if the voucher was acceptable. Boolean values are acceptable, where "true" indicates the voucher was acceptable.

If the voucher was not acceptable the Reason string indicates why. In the failure case this message may be sent to an unauthenticated, potentially malicious registrar and therefore the Reason string SHOULD NOT provide information beneficial to an attacker. The operational benefit of this telemetry information is balanced against the operational costs of not recording that an voucher was ignored by a client the registrar expected to continue joining the domain.

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

The version and status fields MUST be present. The Reason field SHOULD be present whenever the status field is false. The Reason-Context field is optional. In the case of a SUCCESS the Reason string MAY be omitted.

The keys to this JSON object are case-sensitive and MUST be lowercase. Figure 16 shows an example JSON.

```
<CODE BEGINS> file "voucherstatus.cddl"
voucherstatus-post = {
  "version": uint,
  "status": bool,
  ? "reason": text,
  ? "reason-context" : { $$arbitrary-map }
}
<CODE ENDS>
```

Figure 15: CDDL for voucher status POST

```
{
  "version": 1,
  "status":false,
  "reason":"Informative human readable message",
  "reason-context": { "additional" : "JSON" }
}
```

Figure 16: Example Status Telemetry

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

Additional standard JSON fields in this POST MAY be added, see Section 8.5. A server that sees unknown fields should log them, but otherwise ignore them.

5.8. Registrar audit-log request

After receiving the pledge status telemetry Section 5.7, the registrar SHOULD request the MASA audit-log from the MASA service.

This is done with an HTTP POST using the operation path value of `"/.well-known/brski/requestauditlog"`.

The registrar SHOULD HTTP POST the same registrar voucher-request as it did when requesting a voucher (using the same Content-Type). It is posted to the `/requestauditlog` URI instead. The `"idevid-issuer"`

and "serial-number" informs the MASA which log is requested so the appropriate log can be prepared for the response. Using the same media type and message minimizes cryptographic and message operations although it results in additional network traffic. The relying MASA implementation MAY leverage internal state to associate this request with the original, and by now already validated, voucher-request so as to avoid an extra crypto validation.

A registrar MAY request logs at future times. If the registrar generates a new request then the MASA is forced to perform the additional cryptographic operations to verify the new request.

A MASA that receives a request for a device that does not exist, or for which the requesting owner was never an owner returns an HTTP 404 ("Not found") code.

It is reasonable for a Registrar, that the MASA does not believe to be the current owner, to request the audit-log. There are probably reasons for this which are hard to predict in advance. For instance, such a registrar may not be aware that the device has been resold; it may be that the device has been resold inappropriately, and this is how the original owner will learn of the occurrence. It is also possible that the device legitimately spends time in two different networks.

Rather than returning the audit-log as a response to the POST (with a return code 200), the MASA MAY instead return a 201 ("Created") response ([RFC7231] sections 6.3.2 and 7.1), with the URL to the prepared (and idempotent, therefore cachable) audit response in the Location: header field.

In order to avoid enumeration of device audit-logs, MASA that return URLs SHOULD take care to make the returned URL unguessable. [W3C.WD-capability-urls-20140218] provides very good additional guidance. For instance, rather than returning URLs containing a database number such as <https://example.com/auditlog/1234> or the EUI of the device such as <https://example.com/auditlog/10-00-00-11-22-33>, the MASA SHOULD return a randomly generated value (a "slug" in web parlance). The value is used to find the relevant database entry.

A MASA that returns a code 200 MAY also include a Location: header for future reference by the registrar.

5.8.1. MASA audit log response

A log data file is returned consisting of all log entries associated with the device selected by the IDevID presented in the request. The audit log may be abridged by removal of old or repeated values as explained below. The returned data is in JSON format ([RFC8259]), and the Content-Type SHOULD be "application/json".

The following CDDL ([RFC8610]) explains the structure of the JSON format audit-log response:

```
<CODE BEGINS> file "auditlog.cddl"
audit-log-response = {
  "version": uint,
  "events": [ + event ]
  "truncation": {
    ? "nonced duplicates": uint,
    ? "nonceless duplicates": uint,
    ? "arbitrary": uint,
  }
}

event = {
  "date": text,
  "domainID": text,
  "nonce": text / null,
  "assertion": "verified" / "logged" / "proximity",
  ? "truncated": uint,
}
<CODE ENDS>
```

Figure 17: CDDL for audit-log response

An example:

```

{
  "version": "1",
  "events": [
    {
      "date": "2019-05-15T17:25:55.644-04:00",
      "domainID": "BduJhdHPpfhQLyponf48JzXSGZ8=",
      "nonce": "VOUFT-WwrEv0NuAQEHoV7Q",
      "assertion": "proximity",
      "truncated": "0"
    },
    {
      "date": "2017-05-15T17:25:55.644-04:00",
      "domainID": "BduJhdHPpfhQLyponf48JzXSGZ8=",
      "nonce": "f4G6Vilt8nKo/FieCVgpBg==",
      "assertion": "proximity"
    }
  ],
  "truncation": {
    "nonced duplicates": "0",
    "nonceless duplicates": "1",
    "arbitrary": "2"
  }
}

```

Figure 18: Example of audit-log response

The domainID is a binary SubjectKeyIdentifier value calculated according to Section 5.8.2. It is encoded once in base64 in order to be transported in this JSON container.

The date is in [RFC3339] format, which is consistent with typical JavaScript usage of JSON.

The truncation structure MAY be omitted if all values are zero. Any counter missing from the truncation structure is to be assumed to be zero.

The nonce is a string, as provided in the voucher-request, and used in the voucher. If no nonce was placed in the resulting voucher, then a value of null SHOULD be used in preference to omitting the entry. While the nonce is often created as a base64 encoded random series of bytes, this should not be assumed.

Distribution of a large log is less than ideal. This structure can be optimized as follows: Nonced or Nonceless entries for the same domainID MAY be abridged from the log leaving only the single most recent nonced or nonceless entry for that domainID. In the case of truncation the 'event' truncation value SHOULD contain a count of the

number of events for this domainID that were omitted. The log SHOULD NOT be further reduced but there could exist operational situation where maintaining the full log is not possible. In such situations the log MAY be arbitrarily abridged for length, with the number of removed entries indicated as 'arbitrary'.

If the truncation count exceeds 1024 then the MASA MAY use this value without further incrementing it.

A log where duplicate entries for the same domain have been omitted ("nonced duplicates" and/or "nonceless duplicates) could still be acceptable for informed decisions. A log that has had "arbitrary" truncations is less acceptable but manufacturer transparency is better than hidden truncations.

A registrar that sees a version value greater than 1 indicates an audit log format that has been enhanced with additional information. No information will be removed in future versions; should an incompatible change be desired in the future, then a new HTTP end point will be used.

This document specifies a simple log format as provided by the MASA service to the registrar. This format could be improved by distributed consensus technologies that integrate vouchers with technologies such as block-chain or hash trees or optimized logging approaches. Doing so is out of the scope of this document but is an anticipated improvement for future work. As such, the registrar SHOULD anticipate new kinds of responses, and SHOULD provide operator controls to indicate how to process unknown responses.

5.8.2. Calculation of domainID

The domainID is a binary value (a BIT STRING) that uniquely identifies a Registrar by the "pinned-domain-cert".

If the "pinned-domain-cert" certificate includes the SubjectKeyIdentifier (Section 4.2.1.2 [RFC5280]), then it is to be used as the domainID. If not, the SPKI Fingerprint as described in [RFC7469] section 2.4 is to be used. This value needs to be calculated by both MASA (to populate the audit-log), and by the Registrar (to recognize itself in the audit log).

[RFC5280] section 4.2.1.2 does not mandate that the SubjectKeyIdentifier extension be present in non-CA certificates. It is RECOMMENDED that Registrar certificates (even if self-signed), always include the SubjectKeyIdentifier to be used as a domainID.

The domainID is determined from the certificate chain associated with the pinned-domain-cert and is used to update the audit-log.

5.8.3. Registrar audit log verification

Each time the Manufacturer Authorized Signing Authority (MASA) issues a voucher, it appends details of the assignment to an internal audit log for that device. The internal audit log is processed when responding to requests for details as described in Section 5.8. The contents of the audit log can express a variety of trust levels, and this section explains what kind of trust a registrar can derive from the entries.

While the audit log provides a list of vouchers that were issued by the MASA, the vouchers are issued in response to voucher-requests, and it is the contents of the voucher-requests which determines how meaningful the audit log entries are.

A registrar SHOULD use the log information to make an informed decision regarding the continued bootstrapping of the pledge. The exact policy is out of scope of this document as it depends on the security requirements within the registrar domain. Equipment that is purchased pre-owned can be expected to have an extensive history. The following discussion is provided to help explain the value of each log element:

date: The date field provides the registrar an opportunity to divide the log around known events such as the purchase date. Depending on context known to the registrar or administrator events before/after certain dates can have different levels of importance. For example for equipment that is expected to be new, and thus have no history, it would be a surprise to find prior entries.

domainID: If the log includes an unexpected domainID then the pledge could have imprinted on an unexpected domain. The registrar can be expected to use a variety of techniques to define "unexpected" ranging from white lists of prior domains to anomaly detection (e.g. "this device was previously bound to a different domain than any other device deployed"). Log entries can also be compared against local history logs in search of discrepancies (e.g. "this device was re-deployed some number of times internally but the external audit log shows additional re-deployments our internal logs are unaware of").

nonce: Nonceless entries mean the logged domainID could theoretically trigger a reset of the pledge and then take over management by using the existing nonceless voucher.

assertion: The assertion leaf in the voucher and audit log indicates why the MASA issued the voucher. A "verified" entry means that the MASA issued the associated voucher as a result of positive verification of ownership. However, this entry does not indicate whether the pledge was actually deployed in the prior domain, or not. A "logged" assertion informs the registrar that the prior vouchers were issued with minimal verification. A "proximity" assertion assures the registrar that the pledge was truly communicating with the prior domain and thus provides assurance that the prior domain really has deployed the pledge.

A relatively simple policy is to white list known (internal or external) domainIDs, and require all vouchers to have a nonce. An alternative is to require that all 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.

The client MUST send an indicator to the Registrar about its enrollment status. It does this by using an HTTP POST of a JSON dictionary with the of attributes described below to the new EST endpoint at `"/.well-known/brski/enrollstatus"`. (XXX ?)

When indicating a successful enrollment the client SHOULD first re-establish the EST TLS session using the newly obtained credentials. TLS 1.2 supports doing this in-band, but TLS 1.3 does not. The client SHOULD therefore always close the existing TLS connection, and start a new one.

In the case of a failed enrollment, the client MUST send the telemetry information over the same TLS connection that was used for the enrollment attempt, with a Reason string indicating why the most recent enrollment failed. (For failed attempts, the TLS connection is the most reliable way to correlate server-side information with what the client provides.)

The version and status fields MUST be present. The Reason field SHOULD be present whenever the status field is false. In the case of a SUCCESS the Reason string MAY be omitted.

The reason-context attribute is an arbitrary JSON object (literal value or hash of values) which provides additional information specific to the failure to unroll from this pledge. The contents of this field are not subject to standardization. This is represented by the group-socket "\$\$arbitrary-map" in the CDDL.

In the case of a SUCCESS the Reason string is omitted.

```
<CODE BEGINS> file "enrollstatus.cddl"
enrollstatus-post = {
  "version": uint,
  "status": bool,
  ? "reason": text,
  ? "reason-context" : { $$arbitrary-map }
}
<CODE ENDS>
```

Figure 19: CDDL for enrollment status POST

An example status report can be seen below. It is sent with with the media type: application/json

```
{
  "version": 1,
  "status":true,
  "reason":"Informative human readable message",
  "reason-context": { "additional" : "JSON" }
}
```

Figure 20: Example of enrollment status POST

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

Within the server logs the server MUST capture if this message was received over an TLS session with a matching client certificate.

5.9.5. Multiple certificates

Pledges that require multiple certificates could establish direct EST connections to the registrar.

5.9.6. EST over CoAP

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

6. Clarification of transfer-encoding

[RFC7030] defines its endpoints to include a "Content-Transfer-Encoding" heading, and the payloads to be [RFC4648] Base64 encoded DER.

When used within BRSKI, the original RFC7030 EST endpoints remain Base64 encoded, but the new BRSKI end points which send and receive binary artifacts (specifically, `"/.well-known/brski/requestvoucher"`) are binary. That is, no encoding is used.

In the BRSKI context, the EST "Content-Transfer-Encoding" header field if present, SHOULD be ignored. This header field does not need to be included.

7. Reduced security operational modes

A common requirement of bootstrapping is to support less secure operational modes for support specific use cases. This section suggests a range of mechanisms that would alter the security assurance of BRSKI to accommodate alternative deployment architectures and mitigate lifecycle management issues identified in Section 10. They are presented here as informative (non-normative) design guidance for future standardization activities. Section 9 provides standardization applicability statements for the ANIMA ACP. Other users would be expected that subsets of these mechanisms could be profiled with an accompanying applicability statements similar to the one described in Section 9.

This section is considered non-normative in the generality of the protocol. Use of the suggested mechanisms here MUST be detailed in specific profiles of BRSKI, such as in Section 9.

7.1. Trust Model

This section explains the trust relationships detailed in Section 2.4:



Figure 10

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

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

Registrar: When interacting with a MASA a registrar makes all decisions. For Ownership Audit Vouchers (see [RFC8366]) the registrar is provided an opportunity to accept MASA decisions.

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

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

7.2. Pledge security reductions

The following is a list of alternative behaviours that the pledge can be programmed to implement. These behaviours are not mutually exclusive, nor are they dependent upon each other. Some of these methods enable offline and emergency (touch based) deployment use cases. Normative language is used as these behaviours are referenced in later sections in a normative fashion.

1. The pledge **MUST** accept nonceless vouchers. This allows for a use case where the registrar can not connect to the MASA at the deployment time. Logging and validity periods address the security considerations of supporting these use cases.

2. Many devices already support "trust on first use" for physical interfaces such as console ports. This document does not change that reality. Devices supporting this protocol MUST NOT support "trust on first use" on network interfaces. This is because "trust on first use" over network interfaces would undermine the logging based security protections provided by this specification.
3. The pledge MAY have an operational mode where it skips voucher validation one time. For example if a physical button is depressed during the bootstrapping operation. This can be useful if the manufacturer service is unavailable. This behavior SHOULD be available via local configuration or physical presence methods (such as use of a serial/craft console) to ensure new entities can always be deployed even when autonomic methods fail. This allows for unsecured imprint.
4. A craft/serial console could include a command such as "est-enroll [2001:db8:0:1]:443" that begins the EST process from the point after the voucher is validated. This process SHOULD include server certificate verification using an on-screen fingerprint.

It is RECOMMENDED that "trust on first use" or any method of skipping voucher validation (including use of craft serial console) only be available if hardware assisted Network Endpoint Assessment (NEA: [RFC5209]) is supported. This recommendation ensures that domain network monitoring can detect inappropriate use of offline or emergency deployment procedures when voucher-based bootstrapping is not used.

7.3. Registrar security reductions

A registrar can choose to accept devices using less secure methods. They MUST NOT be the default behavior. These methods may be acceptable in situations where threat models indicate that low security is adequate. This includes situations where security decisions are being made by the local administrator:

1. A registrar MAY choose to accept all devices, or all devices of a particular type, at the administrator's discretion. This could occur when informing all registrars of unique identifiers of new entities might be operationally difficult.
2. A registrar MAY choose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This could occur when the pledge does not include an X.509 IDevID factory installed credential. New Entities without

an X.509 IDevID credential MAY form the Section 5.2 request using the Section 5.5 format to ensure the pledge's serial number information is provided to the registrar (this includes the IDevID AuthorityKeyIdentifier value, which would be statically configured on the pledge.) The pledge MAY refuse to provide a TLS client certificate (as one is not available.) The pledge SHOULD support HTTP-based or certificate-less TLS authentication as described in EST RFC7030 section 3.3.2. A registrar MUST NOT accept unauthenticated New Entities unless it has been configured to do so by an administrator that has verified that only expected new entities can communicate with a registrar (presumably via a physically secured perimeter.)

3. A registrar MAY submit a nonceless voucher-requests to the MASA service (by not including a nonce in the voucher-request.) The resulting vouchers can then be stored by the registrar until they are needed during bootstrapping operations. This is for use cases where the target network is protected by an air gap and therefore cannot contact the MASA service during pledge deployment.
4. A registrar MAY ignore unrecognized nonceless log entries. This could occur when used equipment is purchased with a valid history being deployed in air gap networks that required offline vouchers.
5. A registrar MAY accept voucher formats of future types that can not be parsed by the Registrar. This reduces the Registrar's visibility into the exact voucher contents but does not change the protocol operations.

7.4. MASA security reductions

Lower security modes chosen by the MASA service affect all device deployments unless the lower-security behavior is tied to specific device identities. The modes described below can be applied to specific devices via knowledge of what devices were sold. They can also be bound to specific customers (independent of the device identity) by authenticating the customer's Registrar.

7.4.1. Issuing Nonceless vouchers

A MASA has the option of not including a nonce in the voucher, and/or not requiring one to be present in the voucher-request. This results in distribution of a voucher that may never expire and in effect makes the specified Domain an always trusted entity to the pledge during any subsequent bootstrapping attempts. That a nonceless voucher was issued is captured in the log information so that the

registrar can make appropriate security decisions when a pledge joins the Domain. Nonceless vouchers are useful to support use cases where registrars might not be online during actual device deployment.

While a nonceless voucher may include an expiry date, a typical use for a nonceless voucher is for it to be long-lived. If the device can be trusted to have an accurate clock (the MASA will know), then a nonceless voucher CAN be issued with a limited lifetime.

A more typical case for a nonceless voucher is for use with offline onboarding scenarios where it is not possible to pass a fresh voucher-request to the MASA. The use of a long-lived voucher also eliminates concern about the availability of the MASA many years in the future. Thus many nonceless vouchers will have no expiry dates.

Thus, the long lived nonceless voucher does not require the proof that the device is online. Issuing such a thing is only accepted when the registrar is authenticated by the MASA and the MASA is authorized to provide this functionality to this customer. The MASA is RECOMMENDED to use this functionality only in concert with an enhanced level of ownership tracking, the details of which are out of scope for this document.

If the pledge device is known to have a real-time-clock that is set from the factory, use of a voucher validity period is RECOMMENDED.

7.4.2. Trusting Owners on First Use

A MASA has the option of not verifying ownership before responding with a voucher. This is expected to be a common operational model because doing so relieves the manufacturer providing MASA services from having to track ownership during shipping and supply chain and allows for a very low overhead MASA service. A registrar uses the audit log information as a defense in depth strategy to ensure that this does not occur unexpectedly (for example when purchasing new equipment the registrar would throw an error if any audit log information is reported.) The MASA SHOULD verify the 'prior-signed-voucher-request' information for pledges that support that functionality. This provides a proof-of-proximity check that reduces the need for ownership verification. The proof-of-proximity comes from the assumption that the pledge and Join Proxy are on the same link-local connection.

A MASA that practices Trust-on-First-Use (TOFU) for Registrar identity may wish to annotate the origin of the connection by IP address or netblock, and restrict future use of that identity from other locations. A MASA that does this SHOULD take care to not create nuisance situations for itself when a customer has multiple registrars, or uses outgoing IPv4 NAT44 connections that change frequently.

7.4.3. Updating or extending voucher trust anchors

This section deals with the problem of a MASA that is no longer available due to a failed business, or the situation where a MASA is uncooperative to a secondary sale.

A manufacturer could offer a management mechanism that allows the list of voucher verification trust anchors to be extended. [I-D.ietf-netconf-keystore] is one such interface that could be implemented using YANG. Pretty much any configuration mechanism used today could be extended to provide the needed additional update. A manufacturer could even decide to install the domain CA trust anchors received during the EST "cacerts" step as voucher verification anchors. Some additional signals will be needed to clearly identify which keys have voucher validation authority from among those signed by the domain CA. This is future work.

With the above change to the list of anchors, vouchers can be issued by an alternate MASA. This could be the previous owner (the seller), or some other trusted third party who is mediating the sale. If it was a third party, then the seller would need to have taken steps to introduce the third party configuration to the device prior disconnection. The third party (e.g. a wholesaler of used equipment) could however use a mechanism described in Section 7.2 to take control of the device after receiving it physically. This would permit the third party to act as the MASA for future onboarding actions. As the IDevID certificate probably can not be replaced, the new owner's Registrar would have to support an override of the MASA URL.

To be useful for resale or other transfers of ownership one of two situations will need to occur. The simplest is that the device is not put through any kind of factory default/reset before going through onboarding again. Some other secure, physical signal would be needed to initiate it. This is most suitable for redeploying a device within the same Enterprise. This would entail having previous configuration in the system until entirely replaced by the new owner, and represents some level of risk.

The second mechanism is that there would need to be two levels of factory reset. One would take the system back entirely to manufacturer state, including removing any added trust anchors, and the second (more commonly used) one would just restore the configuration back to a known default without erasing trust anchors. This weaker factory reset might leave valuable credentials on the device and this may be unacceptable to some owners.

As a third option, the manufacturer's trust anchors could be entirely overwritten with local trust anchors. A factory default would never restore those anchors. This option comes with a lot of power, but also a lot of responsibility: if access to the private part of the new anchors are lost the manufacturer may be unable to help.

8. IANA Considerations

This document requires the following IANA actions:

8.1. The IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. IANA is asked to register the following:

```
URI: urn:ietf:params:xml:ns:yang:ietf-voucher-request
Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

8.2. YANG Module Names Registry

This document registers a YANG module in the "YANG Module Names" registry [RFC6020]. IANA is asked to register the following:

```
name:          ietf-voucher-request
namespace:     urn:ietf:params:xml:ns:yang:ietf-voucher-request
prefix:        vch
reference:     THIS DOCUMENT
```

8.3. BRSKI well-known considerations

8.3.1. BRSKI .well-known registration

To the Well-Known URIs Registry, at:
"https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml", this document registers the well-known name "brski" with the following filled-in template from [RFC5785]:

```
URI suffix: brski
Change Controller: IETF
```

IANA is asked to change the registration of "est" to now only include RFC7030 and no longer this document. Earlier versions of this document used "/.well-known/est" rather than "/.well-known/brski".

8.3.2. BRSKI .well-known registry

IANA is requested to create a new Registry entitled: "BRSKI well-known URIs". The registry shall have at least three columns: URI, description, and reference. New items can be added using the Specification Required process. The initial contents of this registry shall be:

	URI	document	description
to MASA	requestvoucher	[THISRFC]	pledge to registrar, and from registrar
	voucher_status	[THISRFC]	pledge to registrar
	requestauditlog	[THISRFC]	registrar to MASA
	enrollstatus	[THISRFC]	pledge to registrar

8.4. PKIX Registry

IANA is requested to register the following:

This document requests a number for id-mod-MASAURLExtn2016(TBD) from the pkix(7) id-mod(0) Registry.

This document has received an early allocation from the id-pe registry (SMI Security for PKIX Certificate Extension) for id-pe-masa-url with the value 32, resulting in an OID of 1.3.6.1.5.5.7.1.32.

8.5. Pledge BRSKI Status Telemetry

IANA is requested to create a new Registry entitled: "BRSKI Parameters", and within that Registry to create a table called: "Pledge BRSKI Status Telemetry Attributes". New items can be added using the Specification Required process. The following items are to be in the initial registration, with this document (Section 5.7) as the reference:

- * version
- * Status
- * Reason
- * reason-context

8.6. DNS Service Names

IANA is requested to register the following Service Names:

Service Name: brski-proxy
Transport Protocol(s): tcp
Assignee: IESG <iesg@ietf.org>.
Contact: IESG <iesg@ietf.org>
Description: The Bootstrapping Remote Secure Key
 Infrastructures Proxy
Reference: [This document]

Service Name: brski-registrar
Transport Protocol(s): tcp
Assignee: IESG <iesg@ietf.org>.
Contact: IESG <iesg@ietf.org>
Description: The Bootstrapping Remote Secure Key
 Infrastructures Registrar
Reference: [This document]

8.7. GRASP Objective Names

IANA is requested to register the following GRASP Objective Names:

The IANA is requested to register the value "AN_Proxy" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 4.1.1.

The IANA is requested to register the value "AN_join_registrar" (without quotes) to the GRASP Objectives Names Table in the GRASP Parameter Registry. The specification for this value is this document, Section 4.3.

9. Applicability to the Autonomic Control Plane (ACP)

This document provides a solution to the requirements for secure bootstrap set out in Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance [RFC8368], A Reference Model for Autonomic Networking [I-D.ietf-anima-reference-model] and specifically the An Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane], section 3.2 (Secure Bootstrap), and section 6.1 (ACP Domain, Certificate and Network).

The protocol described in this document has appeal in a number of other non-ANIMA use cases. Such uses of the protocol will be deploying into other environments with different tradeoffs of

privacy, security, reliability and autonomy from manufacturers. As such those use cases will need to provide their own applicability statements, and will need to address unique privacy and security considerations for the environments in which they are used.

The autonomic control plane (ACP) that is bootstrapped by the BRSKI protocol is typically used in medium to large Internet Service Provider organizations. Equivalent enterprises that have significant layer-3 router connectivity also will find significant benefit, particularly if the Enterprise has many sites. (A network consisting of primarily layer-2 is not excluded, but the adjacencies that the ACP will create and maintain will not reflect the topology until all devices participate in the ACP).

In the ACP, the Join Proxy is found to be proximal because communication between the pledge and the join proxy is exclusively on IPv6 Link-Local addresses. The proximity of the Join Proxy to the Registrar is validated by the Registrar using ANI ACP IPv6 Unique Local Addresses (ULA). ULAs are not routable over the Internet, so as long as the Join Proxy is operating correctly the proximity assertion is satisfied. Other uses of BRSKI will need make similar analysis if they use proximity assertions.

As specified in the ANIMA charter, this work "...focuses on professionally-managed networks." Such a network has an operator and can do things like install, configure and operate the Registrar function. The operator makes purchasing decisions and is aware of what manufacturers it expects to see on its network.

Such an operator is also capable of performing bootstrapping of a device using a serial-console (craft console). The zero-touch mechanism presented in this and the ACP document [I-D.ietf-anima-autonomic-control-plane] represents a significant efficiency: in particular it reduces the need to put senior experts on airplanes to configure devices in person.

There is a recognition as the technology evolves that not every situation may work out, and occasionally a human may still have to visit. In recognition of this, some mechanisms are presented in Section 7.2. The manufacturer MUST provide at least one of the one-touch mechanisms described that permit enrollment to be proceed without availability of any manufacturer server (such as the MASA).

The BRSKI protocol is going into environments where there have already been quite a number of vendor proprietary management systems. Those are not expected to go away quickly, but rather to leverage the secure credentials that are provisioned by BRSKI. The connectivity requirements of said management systems are provided by the ACP.

9.1. Operational Requirements

This section collects operational requirements based upon the three roles involved in BRSKI: The Manufacturer Authorized Signing Authority (MASA), the (Domain) Owner and the Device. It should be recognized that the manufacturer may be involved in two roles, as it creates the software/firmware for the device, and also may be the operator of the MASA.

The requirements in this section are presented using BCP14 ([RFC2119], [RFC8174]) language. These do not represent new normative statements, just a review of a few such things in one place by role. They also apply specifically to the ANIMA ACP use case. Other use cases likely have similar, but MAY have different requirements.

9.1.1. MASA Operational Requirements

The manufacturer **MUST** arrange for an online service to be available called the MASA. It **MUST** be available at the URL which is encoded in the IDevID certificate extensions described in Section 2.3.2.

The online service **MUST** have access to a private key with which to sign [RFC8366] format voucher artifacts. The public key, certificate, or certificate chain **MUST** be built in to the device as part of the firmware.

It is **RECOMMENDED** that the manufacturer arrange for this signing key (or keys) to be escrowed according to typical software source code escrow practices [softwareescrow].

The MASA accepts voucher requests from Domain Owners according to an operational practice appropriate for the device. This can range from any domain owner (first-come first-served, on a TOFU-like basis), to full sales channel integration where Domain Owners need to be positively identified by TLS Client Certificate pinned, or HTTP Authentication process. The MASA creates signed voucher artifacts according to its internally defined policies.

The MASA **MUST** operate an audit log for devices that is accessible. The audit log is designed to be easily cacheable and the MASA **MAY** find it useful to put this content on a CDN.

9.1.2. Domain Owner Operational Requirements

The domain owner MUST operate an EST ([RFC7030]) server with the extensions described in this document. This is the JRC or Registrar. This JRC/EST server MUST announce itself using GRASP within the ACP. This EST server will typically reside with the Network Operations Center for the organization.

The domain owner MAY operate an internal certificate authority (CA) that is separate from the EST server, or it MAY combine all activities into a single device. The determination of the architecture depends upon the scale and resiliency requirements of the organization. Multiple JRC instances MAY be announced into the ACP from multiple locations to achieve an appropriate level of redundancy.

In order to recognize which devices and which manufacturers are welcome on the domain owner's network, the domain owner SHOULD maintain a white list of manufacturers. This MAY extend to integration with purchasing departments to know the serial numbers of devices.

The domain owner SHOULD use the resulting overlay ACP network to manage devices, replacing legacy out-of-band mechanisms.

The domain owner SHOULD operate one or more EST servers which can be used to renew the domain certificates (LDevIDs) which are deployed to devices. These servers MAY be the same as the JRC, or MAY be a distinct set of devices, as appropriate for resiliency.

The organization MUST take appropriate precautions against loss of access to the certificate authority private key. Hardware security modules and/or secret splitting are appropriate.

9.1.3. Device Operational Requirements

Devices MUST come with built-in trust anchors that permit the device to validate vouchers from the MASA.

Device MUST come with (unique, per-device) IDevID certificates that include their serial numbers, and the MASA URL extension.

Devices are expected to find Join Proxies using GRASP, and then connect to the JRC using the protocol described in this document.

Once a domain owner has been validated with the voucher, devices are expected to enroll into the domain using EST. Devices are then expected to form ACPs using IPsec over IPv6 Link-Local addresses as described in [I-D.ietf-anima-autonomic-control-plane].

Once a device has been enrolled it SHOULD listen for the address of the JRC using GRASP, and it SHOULD enable itself as a Join Proxy, and announce itself on all links/interfaces using GRASP DULL.

Devices are expected to renew their certificates before they expire.

10. Privacy Considerations

10.1. MASA audit log

The MASA audit log includes the domainID for each domain a voucher has been issued to. This information is closely related to the actual domain identity. A MASA may need additional defenses against Denial of Service attacks (Section 11.1), and this may involve collecting additional (unspecified here) information. This could provide sufficient information for the MASA service to build a detailed understanding the devices that have been provisioned within a domain.

There are a number of design choices that mitigate this risk. The domain can maintain some privacy since it has not necessarily been authenticated and is not authoritatively bound to the supply chain.

Additionally the domainID captures only the unauthenticated subject key identifier of the domain. A privacy sensitive domain could theoretically generate a new domainID for each device being deployed. Similarly a privacy sensitive domain would likely purchase devices that support proximity assertions from a manufacturer that does not require sales channel integrations. This would result in a significant level of privacy while maintaining the security characteristics provided by Registrar based audit log inspection.

10.2. What BRSKI-EST reveals

During the provisional phase of the BRSKI-EST connection between the Pledge and the Registrar, each party reveals its certificates to each other. For the Pledge, this includes the serialNumber attribute, the MASA URL, and the identity that signed the IDevID certificate.

TLS 1.2 reveals the certificate identities to on-path observers, including the Join Proxy.

TLS 1.3 reveals the certificate identities only to the end parties, but as the connection is provisional, an on-path attacker (MITM) can see the certificates. This includes not just malicious attackers, but also Registrars that are visible to the Pledge, but which are not part of the intended domain.

The certificate of the Registrar is rather arbitrary from the point of view of the BRSKI protocol. As no [RFC6125] validations are expected to be done, the contents could be easily pseudonymized. Any device that can see a join proxy would be able to connect to the Registrar and learn the identity of the network in question. Even if the contents of the certificate are pseudonymized, it would be possible to correlate different connections in different locations belong to the same entity. This is unlikely to present a significant privacy concern to ANIMA ACP uses of BRSKI, but may be a concern to other users of BRSKI.

The certificate of the Pledge could be revealed by a malicious Join Proxy that performed a MITM attack on the provisional TLS connection. Such an attacker would be able to reveal the identity of the Pledge to third parties if it chose to so.

Research into a mechanism to do multi-step, multi-party authenticated key agreement, incorporating some kind of zero-knowledge proof would be valuable. Such a mechanism would ideally avoid disclosing identities until pledge, registrar and MASA agree to the transaction. Such a mechanism would need to discover the location of the MASA without knowing the identity of the pledge, or the identity of the MASA. This part of the problem may be unsolvable.

10.3. What BRSKI-MASA reveals to the manufacturer

With consumer-oriented devices, the "call-home" mechanism in IoT devices raises significant privacy concerns. See [livingwithIoT] and [IoTstrangeThings] for exemplars. The Autonomic Control Plane (ACP) usage of BRSKI is not targeted at individual usage of IoT devices, but rather at the Enterprise and ISP creation of networks in a zero-touch fashion where the "call-home" represents a different class of privacy and lifecycle management concerns.

It needs to be re-iterated that the BRSKI-MASA mechanism only occurs once during the commissioning of the device. It is well defined, and although encrypted with TLS, it could in theory be made auditable as the contents are well defined. This connection does not occur when the device powers on or is restarted for normal routines. (It is conceivable, but remarkably unusual, that a device could be forced to go through a full factory reset during an exceptional firmware update situation, after which enrollment would have to be repeated, and a new connection would occur)

The BRSKI call-home mechanism is mediated via the owner's Registrar, and the information that is transmitted is directly auditable by the device owner. This is in stark contrast to many "call-home" protocols where the device autonomously calls home and uses an undocumented protocol.

While the contents of the signed part of the pledge voucher request can not be changed, they are not encrypted at the registrar. The ability to audit the messages by the owner of the network is a mechanism to defend against exfiltration of data by a nefarious pledge. Both are, to re-iterate, encrypted by TLS while in transit.

The BRSKI-MASA exchange reveals the following information to the manufacturer:

- * the identity of the device being enrolled. This is revealed by transmission of a signed voucher-request containing the serial-number. The manufacturer can usually link the serial number to a device model.
- * an identity of the domain owner in the form of the domain trust anchor. However, this is not a global PKI anchored name within the WebPKI, so this identity could be pseudonymous. If there is sales channel integration, then the MASA will have authenticated the domain owner, either via pinned certificate, or perhaps another HTTP authentication method, as per Section 5.5.4.
- * the time the device is activated,
- * the IP address of the domain Owner's Registrar. For ISPs and Enterprises, the IP address provides very clear geolocation of the owner. No amount of IP address privacy extensions ([RFC4941]) can do anything about this, as a simple whois lookup likely identifies the ISP or Enterprise from the upper bits anyway. A passive attacker who observes the connection definitely may conclude that the given enterprise/ISP is a customer of the particular equipment vendor. The precise model that is being enrolled will remain private.

Based upon the above information, the manufacturer is able to track a specific device from pseudonymous domain identity to the next pseudonymous domain identity. If there is sales-channel integration, then the identities are not pseudonymous.

The manufacturer knows the IP address of the Registrar, but it can not see the IP address of the device itself. The manufacturer can not track the device to a detailed physical or network location, only to the location of the Registrar. That is likely to be at the Enterprise or ISPs headquarters.

The above situation is to be distinguished from a residential/individual person who registers a device from a manufacturer. Individuals do not tend to have multiple offices, and their registrar is likely on the same network as the device. A manufacturer that sells switching/routing products to enterprises should hardly be surprised if additional purchases switching/routing products are made. Deviations from a historical trend or an establish baseline would, however, be notable.

The situation is not improved by the enterprise/ISP using anonymization services such as ToR [Dingledine2004], as a TLS 1.2 connection will reveal the ClientCertificate used, clearly identifying the enterprise/ISP involved. TLS 1.3 is better in this regard, but an active attacker can still discover the parties involved by performing a Man-In-The-Middle-Attack on the first attempt (breaking/killing it with a TCP RST), and then letting subsequent connection pass through.

A manufacturer could attempt to mix the BRSKI-MASA traffic in with general traffic their site by hosting the MASA behind the same (set) of load balancers that the companies normal marketing site is hosted behind. This makes lots of sense from a straight capacity planning point of view as the same set of services (and the same set of Distributed Denial of Service mitigations) may be used. Unfortunately, as the BRSKI-MASA connections include TLS ClientCertificate exchanges, this may easily be observed in TLS 1.2, and a traffic analysis may reveal it even in TLS 1.3. This does not make such a plan irrelevant. There may be other organizational reasons to keep the marketing site (which is often subject to frequent re-designs, outsourcing, etc.) separate from the MASA, which may need to operate reliably for decades.

10.4. Manufacturers and Used or Stolen Equipment

As explained above, the manufacturer receives information each time that a device which is in factory-default mode does a zero-touch bootstrap, and attempts to enroll into a domain owner's registrar.

The manufacturer is therefore in a position to decline to issue a voucher if it detects that the new owner is not the same as the previous owner.

1. This can be seen as a feature if the equipment is believed to have been stolen. If the legitimate owner notifies the manufacturer of the theft, then when the new owner brings the device up, if they use the zero-touch mechanism, the new (illegitimate) owner reveals their location and identity.
2. In the case of Used equipment, the initial owner could inform the manufacturer of the sale, or the manufacturer may just permit resales unless told otherwise. In which case, the transfer of ownership simply occurs.
3. A manufacturer could however decide not to issue a new voucher in response to a transfer of ownership. This is essentially the same as the stolen case, with the manufacturer having decided that the sale was not legitimate.
4. There is a fourth case, if the manufacturer is providing protection against stolen devices. The manufacturer then has a responsibility to protect the legitimate owner against fraudulent claims that the equipment was stolen. In the absence of such manufacturer protection, such a claim would cause the manufacturer to refuse to issue a new voucher. Should the device go through a deep factory reset (for instance, replacement of a damaged main board component, the device would not bootstrap.
5. Finally, there is a fifth case: the manufacturer has decided to end-of-line the device, or the owner has not paid a yearly support amount, and the manufacturer refuses to issue new vouchers at that point. This last case is not new to the industry: many license systems are already deployed that have significantly worse effect.

This section has outlined five situations in which a manufacturer could use the voucher system to enforce what are clearly license terms. A manufacturer that attempted to enforce license terms via vouchers would find it rather ineffective as the terms would only be enforced when the device is enrolled, and this is not (to repeat), a daily or even monthly occurrence.

10.5. Manufacturers and Grey market equipment

Manufacturers of devices often sell different products into different regional markets. Which product is available in which market can be driven by price differentials, support issues (some markets may require manuals and tech-support to be done in the local language), government export regulation (such as whether strong crypto is permitted to be exported, or permitted to be used in a particular market). When an domain owner obtains a device from a different market (they can be new) and transfers it to a different location, this is called a Grey Market.

A manufacturer could decide not to issue a voucher to an enterprise/ISP based upon their location. There are a number of ways which this could be determined: from the geolocation of the registrar, from sales channel knowledge about the customer, and what products are (un-)available in that market. If the device has a GPS the coordinates of the device could even be placed into an extension of the voucher.

The above actions are not illegal, and not new. Many manufacturers have shipped crypto-weak (exportable) versions of firmware as the default on equipment for decades. The first task of an enterprise/ISP has always been to login to a manufacturer system, show one's "entitlement" (country information, proof that support payments have been made), and receive either a new updated firmware, or a license key that will activate the correct firmware.

BRSKI permits the above process to automated (in an autonomic fashion), and therefore perhaps encourages this kind of differentiation by reducing the cost of doing it.

An issue that manufacturers will need to deal with in the above automated process is when a device is shipped to one country with one set of rules (or laws or entitlements), but the domain registry is in another one. Which rules apply is something will have to be worked out: the manufacturer could come to believe they are dealing with Grey market equipment, when it is simply dealing with a global enterprise.

10.6. Some mitigations for meddling by manufacturers

The most obvious mitigation is not to buy the product. Pick manufacturers that are up-front about their policies, who do not change them gratuitously.

Section 7.4.3 describes some ways in which a manufacturer could provide a mechanism to manage the trust anchors and built-in certificates (IDevID) as an extension. There are a variety of mechanism, and some may take a substantial amount of work to get exactly correct. These mechanisms do not change the flow of the protocol described here, but rather allow the starting trust assumptions to be changed. This is an area for future standardization work.

Replacement of the voucher validation anchors (usually pointing to the original manufacturer's MASA) with those of the new owner permits the new owner to issue vouchers to subsequent owners. This would be done by having the selling (old) owner to run a MASA.

The BRSKI protocol depends upon a trust anchor on the device and an identity on the device. Management of these entities facilitates a few new operational modes without making any changes to the BRSKI protocol. Those modes include: offline modes where the domain owner operates an internal MASA for all devices, resell modes where the first domain owner becomes the MASA for the next (resold-to) domain owner, and services where an aggregator acquires a large variety of devices, and then acts as a pseudonymized MASA for a variety of devices from a variety of manufacturers.

Although replacement of the IDevID is not required for all modes described above, a manufacturers could support such a thing. Some may wish to consider replacement of the IDevID as an indication that the device's warrantee is terminated. For others, the privacy requirements of some deployments might consider this a standard operating practice.

As discussed at the end of Section 5.8.1, new work could be done to use a distributed consensus technology for the audit log. This would permit the audit log to continue to be useful, even when there is a chain of MASA due to changes of ownership.

10.7. Death of a manufacturer

A common concern has been that a manufacturer could go out of business, leaving owners of devices unable to get new vouchers for existing products. Said products might have been previously deployed, but need to be re-initialized, they might have been purchased used, or they might have kept in a warehouse as long-term spares.

The MASA was named the Manufacturer *Authorized* Signing Authority to emphasize that it need not be the manufacturer itself that performs this. It is anticipated that specialist service providers will come

to exist that deal with the creation of vouchers in much the same way that many companies have outsourced email, advertising and janitorial services.

Further, it is expected that as part of any service agreement that the manufacturer would arrange to escrow appropriate private keys such that a MASA service could be provided by a third party. This has routinely been done for source code for decades.

11. Security Considerations

This document details a protocol for bootstrapping that balances operational concerns against security concerns. As detailed in the introduction, and touched on again in Section 7, the protocol allows for reduced security modes. These attempt to deliver additional control to the local administrator and owner in cases where less security provides operational benefits. This section goes into more detail about a variety of specific considerations.

To facilitate logging and administrative oversight, in addition to triggering Registrar verification of MASA logs, the pledge reports on voucher parsing status to the registrar. In the case of a failure, this information is informative to a potentially malicious registrar. This is mandated anyway because of the operational benefits of an informed administrator in cases where the failure is indicative of a problem. The registrar is RECOMMENDED to verify MASA logs if voucher status telemetry is not received.

To facilitate truly limited clients EST RFC7030 section 3.3.2 requirements that the client MUST support a client authentication model have been reduced in Section 7 to a statement that the registrar "MAY" choose to accept devices that fail cryptographic authentication. This reflects current (poor) practices in shipping devices without a cryptographic identity that are NOT RECOMMENDED.

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

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

11.1. Denial of Service (DoS) against MASA

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

The issuance of nonceless vouchers themselves creates a security concern. If the Registrar of a previous domain can intercept protocol communications then it can use a previously issued nonceless voucher to establish management control of a pledge device even after having sold it. This risk is mitigated by recording the issuance of such vouchers in the MASA audit log that is verified by the subsequent Registrar and by Pledges only bootstrapping when in a factory default state. This reflects a balance between enabling MASA independence during future bootstrapping and the security of bootstrapping itself. Registrar control over requesting and auditing nonceless vouchers allows device owners to choose an appropriate balance.

The MASA is exposed to DoS attacks wherein attackers claim an unbounded number of devices. Ensuring a registrar is representative of a valid manufacturer customer, even without validating ownership of specific pledge devices, helps to mitigate this. Pledge signatures on the pledge voucher-request, as forwarded by the registrar in the prior-signed-voucher-request field of the registrar voucher-request, significantly reduce this risk by ensuring the MASA can confirm proximity between the pledge and the registrar making the request. Supply chain integration ("know your customer") is an additional step that MASA providers and device vendors can explore.

11.2. DomainID must be resistant to second-preimage attacks

The domainID is used as the reference in the audit log to the domain. The domainID is expected to be calculated by a hash that is resistant to a second-preimage attack. Such an attack would allow a second registrar to create audit log entries that are fake.

11.3. Availability of good random numbers

The nonce used by the Pledge in the voucher-request SHOULD be generated by a Strong Cryptographic Sequence ([RFC4086] section 6.2). TLS has a similar requirement.

In particular implementations should pay attention to the advance in [RFC4086] section 3, particularly section 3.4. The random seed used by a device at boot MUST be unique across all devices and all bootstraps. Resetting a device to factory default state does not obviate this requirement.

11.4. Freshness in Voucher-Requests

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

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

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

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

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

This flow is mitigated by the intended registrar verifying the audit logs available from the MASA as described in Section 5.8. Rm might chose to collect a voucher-request but wait until after the intended registrar completes the authorization process before submitting it. This pledge voucher-request would be 'stale' in that it has a nonce that no longer matches the internal state of the pledge. In order to successfully use any resulting voucher the Rm would need to remove the stale nonce or anticipate the pledge's future nonce state. Reducing the possibility of this is why the pledge is mandated to generate a strong random or pseudo-random number nonce.

Additionally, in order to successfully use the resulting voucher the Rm would have to attack the pledge and return it to a bootstrapping enabled state. This would require wiping the pledge of current configuration and triggering a re-bootstrapping of the pledge. This is no more likely than simply taking control of the pledge directly but if this is a consideration the target network is RECOMMENDED to take the following steps:

- * Ongoing network monitoring for unexpected bootstrapping attempts by pledges.
- * Retrieval and examination of MASA log information upon the occurrence of any such unexpected events. Rm will be listed in the logs along with nonce information for analysis.

11.5. Trusting manufacturers

The BRSKI extensions to EST permit a new pledge to be completely configured with domain specific trust anchors. The link from built-in manufacturer-provided trust anchors to domain-specific trust anchors is mediated by the signed voucher artifact.

If the manufacturer's IDevID signing key is not properly validated, then there is a risk that the network will accept a pledge that should not be a member of the network. As the address of the manufacturer's MASA is provided in the IDevID using the extension from Section 2.3, the malicious pledge will have no problem collaborating with it's MASA to produce a completely valid voucher.

BRSKI does not, however, fundamentally change the trust model from domain owner to manufacturer. Assuming that the pledge used its IDevID with RFC7030 EST and BRSKI, the domain (registrar) still needs to trust the manufacturer.

Establishing this trust between domain and manufacturer is outside the scope of BRSKI. There are a number of mechanisms that can adopted including:

- * Manually configuring each manufacturer's trust anchor.
- * A Trust-On-First-Use (TOFU) mechanism. A human would be queried upon seeing a manufacturer's trust anchor for the first time, and then the trust anchor would be installed to the trusted store. There are risks with this; even if the key to name mapping is validated using something like the WebPKI, there remains the possibility that the name is a look alike: e.g, dem0.example. vs demO.example.
- * scanning the trust anchor from a QR code that came with the packaging (this is really a manual TOFU mechanism)
- * some sales integration process where trust anchors are provided as part of the sales process, probably included in a digital packing "slip", or a sales invoice.
- * consortium membership, where all manufacturers of a particular device category (e.g, a light bulb, or a cable-modem) are signed by a certificate authority specifically for this. This is done by CableLabs today. It is used for authentication and authorization as part of TR-79: [docsisroot] and [TR069].

The existing WebPKI provides a reasonable anchor between manufacturer name and public key. It authenticates the key. It does not provide a reasonable authorization for the manufacturer, so it is not directly useable on it's own.

11.6. Manufacturer Maintenance of trust anchors

BRSKI depends upon the manufacturer building in trust anchors to the pledge device. The voucher artifact which is signed by the MASA will be validated by the pledge using that anchor. This implies that the manufacturer needs to maintain access to a signing key that the pledge can validate.

The manufacturer will need to maintain the ability to make signatures that can be validated for the lifetime that the device could be onboarded. Whether this onboarding lifetime is less than the device lifetime depends upon how the device is used. An inventory of devices kept in a warehouse as spares might not be onboarded for many decades.

There are good cryptographic hygiene reasons why a manufacturer would not want to maintain access to a private key for many decades. A manufacturer in that situation can leverage a long-term certificate authority anchor, built-in to the pledge, and then a certificate chain may be incorporated using the normal CMS certificate set. This may increase the size of the voucher artifacts, but that is not a significant issues in non-constrained environments.

There are a few other operational variations that manufacturers could consider. For instance, there is no reason that every device need have the same set of trust anchors pre-installed. Devices built in different factories, or on different days, or any other consideration could have different trust anchors built in, and the record of which batch the device is in would be recorded in the asset database. The manufacturer would then know which anchor to sign an artifact against.

Aside from the concern about long-term access to private keys, a major limiting factor for the shelf-life of many devices will be the age of the cryptographic algorithms included. A device produced in 2019 will have hardware and software capable of validating algorithms common in 2019, and will have no defense against attacks (both quantum and von-neuman brute force attacks) which have not yet been invented. This concern is orthogonal to the concern about access to private keys, but this concern likely dominates and limits the lifespan of a device in a warehouse. If any update to firmware to support new cryptographic mechanism were possible (while the device was in a warehouse), updates to trust anchors would also be done at the same time.

The set of standard operating procedures for maintaining high value private keys is well documented. For instance, the WebPKI provides a number of options for audits at [cabforumaudit], and the DNSSEC root operations are well documented at [dnssecroot].

It is not clear if Manufacturers will take this level of precaution, or how strong the economic incentives are to maintain an appropriate level of security.

This next section examines the risk due to a compromised manufacturer IDevID signing key. This is followed by examination of the risk due to a compromised MASA key. The third section sections below examines the situation where MASA web server itself is under attacker control, but that the MASA signing key itself is safe in a not-directly connected hardware module.

11.6.1. Compromise of Manufacturer IDevID signing keys

An attacker that has access to the key that the manufacturer uses to sign IDevID certificates can create counterfeit devices. Such devices can claim to be from a particular manufacturer, but be entirely different devices: Trojan horses in effect.

As the attacker controls the MASA URL in the certificate, the registrar can be convinced to talk to the attackers' MASA. The Registrar does not need to be in any kind of promiscuous mode to be vulnerable.

In addition to creating fake devices, the attacker may also be able to issue revocations for existing certificates if the IDevID certificate process relies upon CRL lists that are distributed.

There does not otherwise seem to be any risk from this compromise to devices which are already deployed, or which are sitting locally in boxes waiting for deployment (local spares). The issue is that operators will be unable to trust devices which have been in an uncontrolled warehouse as they do not know if those are real devices.

11.6.2. Compromise of MASA signing keys

There are two periods of time in which to consider: when the MASA key has fallen into the hands of an attacker, and after the MASA recognizes that the key has been compromised.

11.6.2.1. Attacker opportunities with compromised MASA key

An attacker that has access to the MASA signing key could create vouchers. These vouchers could be for existing deployed devices, or for devices which are still in a warehouse. In order to exploit these vouchers two things need to occur: the device has to go through a factory default boot cycle, and the registrar has to be convinced to contact the attacker's MASA.

If the attacker controls a Registrar which is visible to the device, then there is no difficulty in delivery of the false voucher. A possible practical example of an attack like this would be in a data center, at an ISP peering point (whether a public IX, or a private peering point). In such a situation, there are already cables attached to the equipment that lead to other devices (the peers at the IX), and through those links, the false voucher could be delivered. The difficult part would be get the device put through a factory reset. This might be accomplished through social engineering of data center staff. Most locked cages have ventilation holes, and possibly a long "paperclip" could reach through to depress a factory

reset button. Once such a piece of ISP equipment has been compromised, it could be used to compromise equipment that was connected to (through long haul links even), assuming that those pieces of equipment could also be forced through a factory reset.

The above scenario seems rather unlikely as it requires some element of physical access; but were there a remote exploit that did not cause a direct breach, but rather a fault that resulted in a factory reset, this could provide a reasonable path.

The above deals with ANI uses of BRSKI. For cases where 802.11 or 802.15.4 is involved, the need to connect directly to the device is eliminated, but the need to do a factory reset is not. Physical possession of the device is not required as above, provided that there is some way to force a factory reset. With some consumer devices with low overall implementation quality, the end users might be familiar with needing to reset the device regularly.

The authors are unable to come up with an attack scenario where a compromised voucher signature enables an attacker to introduce a compromised pledge into an existing operator's network. This is the case because the operator controls the communication between Registrar and MASA, and there is no opportunity to introduce the fake voucher through that conduit.

11.6.2.2. Risks after key compromise is known

Once the operator of the MASA realizes that the voucher signing key has been compromised it has to do a few things.

First, it MUST issue a firmware update to all devices that had that key as a trust anchor, such that they will no longer trust vouchers from that key. This will affect devices in the field which are operating, but those devices, being in operation, are not performing onboarding operations, so this is not a critical patch.

Devices in boxes (in warehouses) are vulnerable, and remain vulnerable until patched. An operator would be prudent to unbox the devices, onboard them in a safe environment, and then perform firmware updates. This does not have to be done by the end-operator; it could be done by a distributor that stores the spares. A recommended practice for high value devices (which typically have a <4hr service window) may be to validate the device operation on a regular basis anyway.

If the onboarding process includes attestations about firmware versions, then through that process the operator would be advised to upgrade the firmware before going into production. Unfortunately, this does not help against situations where the attacker operates their own Registrar (as listed above).

[RFC8366] section 6.1 explains the need for short-lived vouchers. The nonce guarantees freshness, and the short-lived nature of the voucher means that the window to deliver a fake voucher is very short. A nonceless, long-lived voucher would be the only option for the attacker, and devices in the warehouse would be vulnerable to such a thing.

A key operational recommendation is for manufacturers to sign nonceless, long-lived vouchers with a different key 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.

11.6.3. Compromise of MASA web service

An attacker that takes over the MASA web service has a number of attacks. The most obvious one is simply to take the database listing customers and devices and to sell this data to other attackers who will now know where to find potentially vulnerable devices.

The second most obvious thing that the attacker can do is to kill the service, or make it operate unreliably, making customers frustrated. This could have a serious affect on ability to deploy new services by customers, and would be a significant issue during disaster recovery.

While the compromise of the MASA web service may lead to the compromise of the MASA voucher signing key, if the signing occurs offboard (such as in a hardware signing module, HSM), then the key may well be safe, but control over it resides with the attacker.

Such an attacker can issue vouchers for any device presently in service. Said device still needs to be convinced to do through a factory reset process before an attack.

If the attacker has access to a key that is trusted for long-lived nonceless vouchers, then they could issue vouchers for devices which are not yet in service. This attack may be very hard to verify and as it would involve doing firmware updates on every device in warehouses (a potentially ruinously expensive process), a manufacturer might be reluctant to admit this possibility.

11.7. YANG Module Security Considerations

As described in the Security Considerations section of [RFC8366] (section 7.4), the YANG module specified in this document defines the schema for data that is subsequently encapsulated by a CMS signed-data content type, as described in Section 5 of [RFC5652]. As such, all of the YANG modeled data is protected from modification.

The use of YANG to define data structures, via the 'yang-data' statement, is relatively new and distinct from the traditional use of YANG to define an API accessed by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. For this reason, these guidelines do not follow template described by Section 3.7 of [RFC8407].

12. Acknowledgements

We would like to thank the various reviewers for their input, in particular William Atwood, Brian Carpenter, Fuyu Eleven, Eliot Lear, Sergey Kasatkin, Anoop Kumar, Tom Petch, Markus Stenberg, Peter van der Stok, and Thomas Werner

Significant reviews were done by Jari Arko, Christian Huitema and Russ Housley.

Henk Birkholz contributed the CDDL for the audit log response.

This document started it's life as a two-page idea from Steinthor Bjarnason.

In addition, significant review comments were received by many IESG members, including Adam Roach, Alexey Melnikov, Alissa Cooper, Benjamin Kaduk, Eric Vyncke, Roman Danyliw, and Magnus Westerlund.

13. References

13.1. Normative References

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

Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", Work in Progress, Internet-Draft, draft-ietf-anima-autonomic-control-plane-30, 30 October 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-autonomic-control-plane-30.txt>>.

[I-D.ietf-anima-grasp]

Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", Work in Progress,

- Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-grasp-15.txt>>.
- [IDevID] "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [ITU.X690.1994] International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.
- [REST] Fielding, R.F., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowitz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <<https://www.rfc-editor.org/info/rfc3748>>.
- [RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <<https://www.rfc-editor.org/info/rfc3927>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.

- [RFC4519] Sciberras, A., Ed., "Lightweight Directory Access Protocol (LDAP): Schema for User Applications", RFC 4519, DOI 10.17487/RFC4519, June 2006, <<https://www.rfc-editor.org/info/rfc4519>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <<https://www.rfc-editor.org/info/rfc5272>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <<https://www.rfc-editor.org/info/rfc7469>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <<https://www.rfc-editor.org/info/rfc7951>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <<https://www.rfc-editor.org/info/rfc8368>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

13.2. Informative References

- [brewski] "Urban Dictionary: Brewski", October 2019, <<https://www.urbandictionary.com/define.php?term=brewski>>.
- [cabforumaudit] "Information for Auditors and Assessors", August 2019, <<https://cabforum.org/information-for-auditors-and-assessors/>>.
- [Dingledine2004] Dingledine, R., Mathewson, N., and P. Syverson, "Tor: the second-generation onion router", 2004, <<https://spec.torproject.org/tor-spec>>.

[dnssecroot]

"DNSSEC Practice Statement for the Root Zone ZSK Operator", December 2017, <<https://www.iana.org/dnssec/dps/zsk-operator/dps-zsk-operator-v2.0.pdf>>.

[docsisroot]

"CableLabs Digital Certificate Issuance Service", February 2018, <<https://www.cablelabs.com/resources/digital-certificate-issuance-service/>>.

[I-D.ietf-ace-coap-est]

Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST over secure CoAP (EST-coaps)", Work in Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6 January 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-ace-coap-est-18.txt>>.

[I-D.ietf-anima-constrained-voucher]

Richardson, M., Stok, P., and P. Kampanakis, "Constrained Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-09, 2 November 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-anima-constrained-voucher-09.txt>>.

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

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

[I-D.ietf-netconf-keystore]

Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-20, 20 August 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-netconf-keystore-20.txt>>.

[I-D.richardson-anima-state-for-joinrouter]

Richardson, M., "Considerations for stateful vs stateless join router in ANIMA bootstrap", Work in Progress, Internet-Draft, draft-richardson-anima-state-for-joinrouter-03, 22 September 2020, <<http://www.ietf.org/internet-drafts/draft-richardson-anima-state-for-joinrouter-03.txt>>.

- [imprinting] "Wikipedia article: Imprinting", July 2015, <[https://en.wikipedia.org/wiki/Imprinting_\(psychology\)](https://en.wikipedia.org/wiki/Imprinting_(psychology))>.
- [IoTstrangeThings] "IoT of toys stranger than fiction: Cybersecurity and data privacy update (accessed 2018-12-02)", March 2017, <<https://www.welivesecurity.com/2017/03/03/internet-of-things-security-privacy-iot-update/>>.
- [livingwithIoT] "What is it actually like to live in a house filled with IoT devices? (accessed 2018-12-02)", February 2018, <<https://www.siliconrepublic.com/machines/iot-smart-devices-reality>>.
- [minerva] Richardsdon, M., "Minerva reference implementation for BRSKI", 2020, <<https://minerva.sandelman.ca/>>.
- [minervagithub] Richardsdon, M., "GITHUB hosting of Minerva reference code", 2020, <<https://github.com/ANIMAgus-minerva>>.
- [openssl] "OpenSSL X509 utility", September 2019, <<https://www.openssl.org/docs/man1.1.1/man1/openssl-x509.html>>.
- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, DOI 10.17487/RFC2663, August 1999, <<https://www.rfc-editor.org/info/rfc2663>>.
- [RFC5209] Sangster, P., Khosravi, H., Mani, M., Narayan, K., and J. Tardo, "Network Endpoint Assessment (NEA): Overview and Requirements", RFC 5209, DOI 10.17487/RFC5209, June 2008, <<https://www.rfc-editor.org/info/rfc5209>>.
- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<https://www.rfc-editor.org/info/rfc5785>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.

- [RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, DOI 10.17487/RFC6961, June 2013, <<https://www.rfc-editor.org/info/rfc6961>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.
- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [slowloris] "Slowloris (computer security)", February 2019, <[https://en.wikipedia.org/wiki/Slowloris_\(computer_security\)](https://en.wikipedia.org/wiki/Slowloris_(computer_security))>.
- [softwareescrow] "Wikipedia article: Software Escrow", October 2019, <https://en.wikipedia.org/wiki/Source_code_escrow>.
- [Stajano99theresurrecting] Stajano, F. and R. Anderson, "The resurrecting duckling: security issues for ad-hoc wireless networks", 1999, <<https://www.cl.cam.ac.uk/~fms27/papers/1999-StajanoAnd-duckling.pdf>>.
- [TR069] "TR-69: CPE WAN Management Protocol", February 2018, <<https://www.broadband-forum.org/standards-and-software/technical-specifications/tr-069-files-tools>>.

[W3C.WD-capability-urls-20140218]

Tennison, J., "Good Practices for Capability URLs", World Wide Web Consortium WD WD-capability-urls-20140218, 18 February 2014, <<https://www.w3.org/TR/2014/WD-capability-urls-20140218>>.

Appendix A. IPv4 and non-ANI operations

The specification of BRSKI in Section 4 intentionally only covers the mechanisms for an IPv6 pledge using Link-Local addresses. This section describes non-normative extensions that can be used in other environments.

A.1. IPv4 Link Local addresses

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

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

A.2. Use of DHCPv4

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

Appendix B. mDNS / DNSSD proxy discovery options

Pledge discovery of the proxy (Section 4.1) MAY be performed with DNS-based Service Discovery [RFC6763] over Multicast DNS [RFC6762] to discover the proxy at "_brski-proxy._tcp.local."

Proxy discovery of the registrar (Section 4.3) MAY be performed with DNS-based Service Discovery over Multicast DNS to discover registrars by searching for the service "_brski-registrar._tcp.local."

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

Discovery of registrar MAY also be performed with DNS-based service discovery by searching for the service "_brski-registrar._tcp.example.com". In this case the domain "example.com" is discovered as described in [RFC6763] section 11 (Appendix A.2 suggests the use of DHCP parameters).

If no local proxy or registrar service is located using the GRASP mechanisms or the above mentioned DNS-based Service Discovery methods, the pledge MAY contact a well known manufacturer provided bootstrapping server by performing a DNS lookup using a well known URI such as "brski-registrar.manufacturer.example.com". The details of the URI are manufacturer specific. Manufacturers that leverage this method on the pledge are responsible for providing the registrar service. Also see Section 2.7.

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

Appendix C. Example Vouchers

Three entities are involved in a voucher: the MASA issues (signs) it, the registrar's public key is mentioned in the voucher, and the pledge validates it. In order to provide reproduceable examples the public and private keys for an example MASA and registrar are first listed.

The keys come from an open source reference implementation of BRSKI, called "Minerva" [minerva]. It is available on github [minervagithub]. The keys presented here are used in the unit and integration tests. The MASA code is called "highway", the Registrar code is called "fountain", and the example client is called "reach".

The public key components of each are presented as both base64 certificates, as well as being decoded by openssl's x509 utility so that the extensions can be seen. This was version 1.1.1c of the [openssl] library and utility.

C.1. Keys involved

The Manufacturer has a Certificate Authority that signs the pledge's IDevID. In addition the Manufacturer's signing authority (the MASA) signs the vouchers, and that certificate must be distributed to the devices at manufacturing time so that vouchers can be validated.

C.1.1. Manufacturer Certificate Authority for IDevID signatures

This private key is Certificate Authority that signs IDevID certificates:

```
<CODE BEGINS> file "vendor.key"
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDcAYkoLWlIEA5SKKhMMdkTK7sJxk5ybKqYq9Yr5aR7tNwqXyLGS7z8G
8S4w/UJ58BqgBwYFK4EEACKhZANiAAQu5/yktJbFLjMC87h7b+yTreFuF8GwewKH
L4mS0r0dVAQubqDUQcTrjvpXrXCpTojiLCzgp8fzkcUDkZ9LD/M90LDipiLNIokP
juF8QkoAbT8pMrY83MS8y76wZ7AalNQ=
-----END EC PRIVATE KEY-----
<CODE ENDS>
```

This public key validates IDevID certificates:

file: examples/vendor.key

```
<CODE BEGINS> file "vendor.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 519772114 (0x1efb17d2)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.ex
ample.com CA
    Validity
      Not Before: Feb 12 22:22:21 2019 GMT
      Not After : Feb 11 22:22:21 2021 GMT
    Subject: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.e
xample.com CA
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (384 bit)
      pub:
        04:2e:e7:fc:a4:b4:96:c5:2e:33:02:f3:b8:7b:6f:
        ec:93:ad:e1:6e:17:c1:b0:7b:02:87:2f:89:92:d2:
        bd:1d:54:04:2e:6e:a0:d4:41:c4:eb:8e:fa:57:ad:
        70:a9:4e:88:e2:2c:2c:e0:a7:c7:f3:91:c5:03:91:
        9f:4b:0f:f3:3d:d0:b0:e2:a6:22:cd:20:e9:0f:8e:
        e1:7c:42:4a:00:6d:3f:29:32:b6:3c:dc:c4:bc:cb:
        be:b0:67:b0:1a:94:d4
      ASN1 OID: secp384r1
      NIST CURVE: P-384
    X509v3 extensions:
      X509v3 Basic Constraints: critical
        CA:TRUE
      X509v3 Key Usage: critical
        Certificate Sign, CRL Sign
      X509v3 Subject Key Identifier:
```



```

<CODE BEGINS> file "masa.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 463036244 (0x1b995f54)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.ex
ample.com CA
    Validity
      Not Before: Feb 12 22:22:41 2019 GMT
      Not After : Feb 11 22:22:41 2021 GMT
    Subject: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.e
xample.com MASA
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:aa:04:15:a3:44:b9:e2:44:f8:c9:f9:1b:07:1b:
        a6:74:73:9c:1e:ba:6c:a9:b3:a9:30:a9:a2:32:59:
        f7:a0:1d:47:01:6d:b9:30:95:7e:82:a8:b8:b4:c1:
        5f:48:9d:22:13:0b:7c:92:cc:df:59:72:b8:ac:b8:
        09:4b:69:a7:a5
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Basic Constraints: critical
      CA:FALSE
    Signature Algorithm: ecdsa-with-SHA256
      30:66:02:31:00:bd:55:e5:9b:0e:fb:fc:5e:95:29:e3:81:b3:
      15:35:aa:93:18:a2:04:be:44:72:b2:51:7d:4d:6d:eb:d1:d5:
      c1:10:3a:b2:39:7b:57:3f:c5:cc:b0:a3:0e:e7:99:46:ba:02:
      31:00:f6:7f:44:7d:b7:14:fa:d1:67:6a:d4:11:c3:4b:ae:e6:
      fb:9a:98:56:fa:85:21:2e:5c:48:4c:f0:3f:f2:9b:3f:ae:88:
      20:a7:ae:f9:72:ff:5b:f9:78:68:cf:0f:48:c9
-----BEGIN CERTIFICATE-----
MIIB3zCCAWSgAwIBAgIEG51fVDAKBggqhkJOPQODAjbDmQ8wDQYDVQQGEWZDYW5h
ZGExEDAOBgNVBAMwMB09udGFyaW8xeEjAQBGNVBAsMVCVhbmRlbG1hbG1hbmR1eE
AwWbaGlnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMB4XDTE5MDIxMjIyMjIyMjIy
DTIxMDIxMTIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy
cm1vMRIwEAYDVQQLDAlTYW5kZWxtYW4xJjAkBgNVBAMMHWhpZ2h3YXktZGVzdC5l
eGFtcGx1LmNvbSBNQVNBMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEqgQVo0S5
4kT4yfkBxumdhOchrpsqbOpMKmiMln3oB1HAW25MJV+gqi4tMFfSJ0iEwt8kszf
WXX4rLgJS2mnpaMQMA4wDAYDVR0TAQH/BAIwADAKBggqhkJOPQODAgNpADBMAjEA
vVX1mw77/F6VKeOBsxU1qpMYogS+RHKyUX1NbevR1cEQOrI5e1c/xcywow7nmUa6
AjEA9n9EfbCU+tFnatQRw0uu5vuamFb6hSEuXEhM8D/ymz+uiCCnrVly/1v5eGjP
D0jJ
-----END CERTIFICATE-----
<CODE ENDS>

```

C.1.3. Registrar Certificate Authority

This Certificate Authority enrolls the pledge once it is authorized, and it also signs the Registrar's certificate.

```
<CODE BEGINS> file "ownerca_secp384rl.key"
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDChnLI0MSOLf8XndiZqoZdqblcPR5YSoPGhPOuFxFxWy1gFi9HbWv8b/R
EGdRgGEVSjKgBwYFK4EEACKhZANiAAQbf1m6F8MavGaNjGzgw/oxcQ919iKRvbdW
gAfb37h6pUVNeYpGlxlZ1jGxj2l9Mr48yD5bY7VG9qjVb5v5wPPTuRQ/ckdRpHbd
0vC/9cqPMAF/+MJf0/UgAOSLi/IHbLQ=
-----END EC PRIVATE KEY-----
<CODE ENDS>
```

The public key is indicated in a pledge voucher-request to show proximity.

file: examples/ownerca_secp384rl.key

```
<CODE BEGINS> file "ownerca_secp384rl.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 694879833 (0x296b0659)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstru
ng Fountain Root CA
    Validity
      Not Before: Feb 25 21:31:45 2020 GMT
      Not After : Feb 24 21:31:45 2022 GMT
    Subject: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstr
ung Fountain Root CA
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (384 bit)
      pub:
        04:1b:7f:59:ba:17:c3:1a:bc:66:8d:8c:6c:e0:c3:
        fa:31:71:0f:65:f6:22:91:bd:b7:56:80:07:db:df:
        b8:7a:a5:45:4d:79:8a:46:97:19:59:96:31:b1:8f:
        69:7d:32:be:3c:c8:3e:5b:63:b5:46:f6:a8:d5:6f:
        9b:f9:c0:f3:d3:b9:14:3f:72:47:51:a4:76:dd:d2:
        f0:bf:f5:ca:8f:30:01:7f:f8:c2:5f:d3:f5:20:03:
        44:8b:8b:f2:07:6c:b4
      ASN1 OID: secp384rl
      NIST CURVE: P-384
    X509v3 extensions:
      X509v3 Basic Constraints: critical
      CA:TRUE
      X509v3 Key Usage: critical
      Certificate Sign, CRL Sign
```

X509v3 Subject Key Identifier:
 B9:A5:F6:CB:11:E1:07:A4:49:2C:A7:08:C6:7C:10:BC:87:B3:74:26
 X509v3 Authority Key Identifier:
 keyid:B9:A5:F6:CB:11:E1:07:A4:49:2C:A7:08:C6:7C:10:BC:87:B3:74

:26

Signature Algorithm: ecdsa-with-SHA256
 30:64:02:30:20:83:06:ce:8d:98:a4:54:7a:66:4c:4a:3a:70:
 c2:52:36:5a:52:8d:59:7d:20:9b:2a:69:14:58:87:38:d8:55:
 79:dd:fd:29:38:95:1e:91:93:76:b4:f5:66:29:44:b4:02:30:
 6f:38:f9:af:12:ed:30:d5:85:29:7c:b1:16:58:bd:67:91:43:
 c4:0d:30:f9:d8:1c:ac:2f:06:dd:bc:d5:06:42:2c:84:a2:04:
 ea:02:a4:5f:17:51:26:fb:d9:2f:d2:5c

```
-----BEGIN CERTIFICATE-----
MIICazCCAFKgAwIBAgIEKWsGWTAKBggqhkJOPQQDAjBtMRIwEAYKczImiZPyLGQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMMM2ZvdW50
YWluLXRlc3QuZXhhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9vdCBDQTAe
Fw0yMDAyMjUyMTMxNDVaFw0yMjUyMTMxNDVaMG0xEjAQBgoJkiaJk/IsZAEZF
FgJjYTEZMBCGcmSjJmT8ixkARKWCXNhbmlbG1hbjeE8MDoGAlUEAwZm91bnRh
aW4tdGVzdC5leGFtcGxlLmNvbSBVbnN0cnVuZyBGb3VudGFpb290IENBMHYw
EAYHkoZiZjOCAQYFK4EEACIDYgAEG39ZuhfDGrxmjYxs4MP6MXEPZfyikb23VoAH
29+4eqVFTXmKRpcZWZYxsY9pfTK+PMg+W201Rvao1W+b+cDz07kUP3JHUaR23dLw
v/XKjzABf/jCX9P1IANEi4vyB2y0o2MwYTAPBgNVHRMBAf8EBTADAQH/MA4GA1Ud
DwEB/wQEAWIBBjAdBgNVHQ4EFgQUuaX2yxHhB6RJLkCIXnwQvIezdCYwHwYDVR0j
BBgwFoAUuaX2yxHhB6RJLkCIXnwQvIezdCYwCgYIKoZIzj0EAwIDZwAwZAIwIIMG
zo2YpFR6ZkxKOnDCUjZaUo1ZfSCbKmkUWIc42FV53f0pOJUekZN2tPvmKUS0AjBv
OPmvEu0w1YUpfLEWWL1nkUPEDTD52BysLwbdvNUGQiyEogTqAqRfF1Em+9kv0lw=
-----END CERTIFICATE-----
<CODE ENDS>
```

C.1.4. Registrar key pair

The Registrar is the representative of the domain owner. This key signs registrar voucher-requests, and terminates the TLS connection from the pledge.

```
<CODE BEGINS> file "jrc_prime256v1.key"
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIFZodk+PC5Mu24+ra0sb0jKzan+dW5rvDAR7YuJUOC1YoAoGCCqGSM49
AwEHoUQDQgAEImlVQcjs6n+Xd5l/28IFv6UiegQwSBztGj5dkK2MAjQIPV8l8lH+E
jLIOYdbJiI0VtEIf1/Jqt+TOBfinTNOLog==
-----END EC PRIVATE KEY-----
<CODE ENDS>
```

The public key is indicated in a pledge voucher-request to show proximity.

```
<CODE BEGINS> file "jrc_prime256v1.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 1066965842 (0x3f989b52)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: DC = ca, DC = sandelman, CN = fountain-test.example.com Unstru
ng Fountain Root CA
    Validity
      Not Before: Feb 25 21:31:54 2020 GMT
      Not After : Feb 24 21:31:54 2022 GMT
    Subject: DC = ca, DC = sandelman, CN = fountain-test.example.com
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:96:65:50:72:34:ba:9f:e5:dd:e6:5f:f6:f0:81:
        6f:e9:48:9e:81:0c:12:07:3b:46:8f:97:64:2b:63:
        00:8d:02:0f:57:c9:7c:94:7f:84:8c:b2:0e:61:d6:
        c9:88:8d:15:b4:42:1f:d7:f2:6a:b7:e4:ce:05:f8:
        a7:4c:d3:8b:3a
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Extended Key Usage: critical
      CMC Registration Authority
      X509v3 Key Usage: critical
      Digital Signature
    Signature Algorithm: ecdsa-with-SHA256
      30:65:02:30:66:4f:60:4c:55:48:1e:96:07:f8:dd:1f:b9:c8:
      12:8d:45:36:87:9b:23:c0:bc:bb:f1:cb:3d:26:15:56:6f:5f:
      1f:bf:d5:1c:0e:6a:09:af:1b:76:97:99:19:23:fd:7e:02:31:
      00:bc:ac:c3:41:b0:ba:0d:af:52:f9:9c:6e:7a:7f:00:1d:23:
      c8:62:01:61:bc:4b:c5:c0:47:99:35:0a:0c:77:61:44:01:4a:
      07:52:70:57:00:75:ff:be:07:0e:98:cb:e5
-----BEGIN CERTIFICATE-----
MIIB/DCCAYKgAwIBAgIEP5ibUjAKBggqhkJOPQQDAjBtMRItwEAYKCZImiZPyLgQB
GRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNVBAMMM2ZvdW50
YWluLXRlc3QuZXRhbXBsZS5jb20gVW5zdHJ1bmcgRm91bnRhaW4gUm9vdCBDQTAe
Fw0yMDAyMjUyMTMxNTRaFw0yMjAyMjUyMTMxNTRaMFMxZjAQBgoJkiaJk/IsZAEZ
FgJjYTEZMBcGCgmSjomT8ixkARKWCXNhbMR1bG1hbJjEiMCAGAlUEAwWZm91bnRha
w4tdGVzdC5leGFtcGxlLmNvbTBZMBMGBYqGSM49AgEGCCqGSM49AwEHA0IABJZl
UHI0up/l3eZf9vCBb+lInoEMEGc7Ro+XZCtjAI0CD1fJfJR/hIyyDmHWyYiNFbRC
H9fyarfkzgzX4p0zTizqjKjAoMByGA1UdJQEB/wQMMAoGCCsGAQUFBwMcMA4GA1Ud
DwEB/wQEAwIHgDAKBggqhkJOPQQDAgNoADB1AjBmT2BMVUgelgf43R+5yBKNRTaH
myPAvLvxyz0mFVZvXx+/1RwOagmvG3aXmRkj/X4CMQC8rMNBsLoNr1L5nG56fwAd
I8hiAWG8S8XAR5k1Cgx3YUQBSgdScFcAdf++Bw6Yy+U=
-----END CERTIFICATE-----
<CODE ENDS>
```

C.1.5. Pledge key pair

The pledge has an IDevID key pair built in at manufacturing time:

```
<CODE BEGINS> file "idevid_00-D0-E5-F2-00-02.key"
-----BEGIN EC PRIVATE KEY-----
MHcCAQEIEIBHNh6r8QRevRuo+tEmBJeFjQKf6bpFA/9NGoltv+9sNoAoGCCqGSM49
AwEHoUQDQgAEA6N1Q4ezfMAKmoecrfb0OBMc1AyEH+BATkF58FsTSyBxs0SbSWLx
FjDOuwB9gLGn2TsTUJumJ6VPw5Z/TP4hJw==
-----END EC PRIVATE KEY-----
<CODE ENDS>
```

The certificate is used by the registrar to find the MASA.

```
<CODE BEGINS> file "idevid_00-D0-E5-F2-00-02.cert"
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 226876461 (0xd85dc2d)
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C = Canada, ST = Ontario, OU = Sandelman, CN = highway-test.ex
ample.com CA
    Validity
      Not Before: Feb  3 06:47:20 2020 GMT
      Not After  : Dec 31 00:00:00 2999 GMT
    Subject: serialNumber = 00-D0-E5-F2-00-02
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:03:a3:75:43:87:b3:7c:c0:0a:9a:87:9c:ad:f6:
        f4:38:13:1c:d4:0c:84:1f:e0:40:4e:41:79:f0:5b:
        13:4b:20:71:b3:44:9b:49:62:f1:16:30:ce:bb:00:
        7d:80:b1:a7:d9:3b:13:50:9b:a6:27:a5:4f:c3:96:
        7f:4c:fe:21:27
      ASN1 OID: prime256v1
      NIST CURVE: P-256
    X509v3 extensions:
      X509v3 Subject Key Identifier:
        45:88:CC:96:96:00:64:37:B0:BA:23:65:64:64:54:08:06:6C:56:AD
      X509v3 Basic Constraints:
        CA:FALSE
      1.3.6.1.5.5.7.1.32:
        ..highway-test.example.com:9443
    Signature Algorithm: ecdsa-with-SHA256
      30:65:02:30:23:e1:a9:2e:ef:22:12:34:5a:a5:c2:15:d6:28:
      bd:ed:3d:96:d6:ce:04:95:ef:a7:c8:dc:18:a8:31:c7:b8:04:
      34:f2:b7:4d:79:8a:67:22:24:03:4f:c5:cd:d6:06:ba:02:31:
      00:b3:8d:5c:0a:d0:fe:04:83:90:d3:4f:6d:72:97:b3:3e:02:
```

```

ea:f1:c8:5a:32:72:58:b7:45:02:50:78:bc:04:1d:23:5e:22:
6f:c3:7f:8c:7c:d7:9b:70:20:91:b4:e1:7f
-----BEGIN CERTIFICATE-----
MIIB5jCCAWygAwIBAgIEDYXcLTAKBggqhkJOPQODAjbDMQ8wDQYDVQQGEwZDYW5h
ZGExEDAOBgNVBAMwMB09udGFyaW8xEjAQBgNVBAsMCVhbmR1bG1hbG1hbjEjMCIGA1UE
AwwbaGlnaHdheS10ZXN0LmV4YW1wbGUuY29tIENBMCAXDTEwMDIwMzA2NDcyMfoY
DzI5OTkxMjMxMDAwMDAwWjAcMR0wGAYDVQQFDBEwMC1EMC1FNs1GMi0wMC0wMjBZ
MBMGByqGSM49AgEGCCqGSM49AwEHA0IABAOjdUOHs3zACpQHnK329DgTHNQmHb/g
QE5BefBbE0sgcbNEM0li8RYwzrsAfYcXp9k7E1CbpielT8OWf0z+ISejWTBxMB0G
A1UdDgQWBBRFiMyWlgBkN7C6I2VkJZFIbmxWrTAJBGNVHRMEAjAAMCsGCCsGAQUF
BwEgBB8MHWhpZ2h3YXktZGVzdC5leGFtcGx1LmNvbTo5NDQzMAoGCCqGSM49BAMC
A2gAMGUCMCPHqS7vIhI0WqXCFdYove09ltbOBjXvp8jcGKgx7gENPK3TXmKZyIk
A0/FzdYGugIxALONXArQ/gSDkNNPbXKXsz4C6vHIWjJyWldFA1B4vAQdI14ib8N/
jHzXm3AgkbThfw==
-----END CERTIFICATE-----
<CODE ENDS>

```

C.2. Example process

The JSON examples below are wrapped at 60 columns. This results in strings that have newlines in them, which makes them invalid JSON as is. The strings would otherwise be too long, so they need to be unwrapped before processing.

For readability, the output of the `asn1parse` has been truncated at 72 columns rather than wrapped.

C.2.1. Pledge to Registrar

As described in Section 5.2, the pledge will sign a pledge voucher-request containing the registrar's public key in the `proximity-registrar-cert` field. The base64 has been wrapped at 60 characters for presentation reasons.

```
<CODE BEGINS> file "vr_00-D0-E5-F2-00-02.b64"
MIIG3wYJKoZIhvcNAQcCoIIIG0DCCBswCAQEExDTALBg1ghkgBZQMEAgEwggOJBgkqhkiG9w0BBwGg
ggN6BIIDdnsiaWV0Zi12b3VjaGVyLXJlcXVlc3Q6dm91Y2h1ciI6eyJhc3N1cnRpb24iOiJwcm94
aW1pdHkiLCJjcmVhdGVkLW9uIjoimjAyMC0wMi0yNVQxODowND00C42NTItMDU6MDAiLCJzZXJp
YWwtbnVtYmVyIjoimDAtRDAtRTUtRjItMDAtMDIiLCJub25jZSI6ImFNamd1ZUtVVC0yMndWaW1q
NnoyN1EiLCJwcm94aW1pdHktcmVnaXN0cmFyLWN1cnQiOiJNSU1CL0RDQ0FzS2dBd0lCQWdJRVA1
aWJVakFLQmdncWhrak9QUVFEQWpCdE1SSXdfQV1LQ1pJbWlaUHM1MR1FCR1JZQ1kyRXhHVEFYQmdv
SmtpYUprL0lZWkFFWkZnbHpZVzVrWld4dFlXNHhQREe2QmdOVkJBtU1NMlp2ZFclMF1XbHVMWFJs
YzNRdVpYaGhiWEJzWlM1amIyMGdWVzV6ZEhKMWJtY2dSbTkxYm5SaGFxNGdVbTl2ZENCRFFUQWVG
dzB5TURBeU1qVXlNVE14TlRSYUZ3MH1NakF5TWpReU1UTXhOVFJhTUZNeEVqQVFCZ29Ka2lhSmsv
SXNaQUVaRmdKallURVpNqMNHQ2dtU0pvbVQ4aXhrQVJrV0NYTmh1bVJsYkcxagJqRW1NQ0FHQTFV
RUF3dlpabTkxYm5SaGFxNHRkR1Z6ZEM1bGVHRnRjR3hsTG1OdmJUQ1pNQ1hQnlxR1NNND1BZ0VH
Q0NxR1NNND1Bd0VIQTBJQUJKWmxVSEkwdXAvbDN1WmY5dkNCYitsSW5vRU1Fz2M3Um8rWFpDdGpB
STBDRDFmSmZKU19oSX15RG1IV3lZaU5GYlJDSdlmeWFyZmt6Z1g0cDB6VG16cWpLakFvTUJZROEx
VWRKUUVCL3dRTU1Bb0dDQ3NHQVFVRkR3JTWNNQTRHQTfVZER3RUIvd1FFQXdJSGdEQUtCZ2dxaGtq
T1BRUURBZ05vQURCbEFqQm1Umk1JN1VnZWxnZjQzUis1eUJLTlJUyUhteVBBdkx2eH16MG1GV1p2
WHgrLzFSd09hZ212RzNhWG1Sa2ovWDRDTRVfDOHJNTkZjZTG9OcJFMNW5HNTZmd0FkSThoaUFXRzhT
OFhBUjVrMUNneDNZVVFcu2dkU2NGY0FkZisrQnc2WXkrVT0ifX2gggHqMIIB5jCAWygAwIBAgIE
DYXcLTAKBggqhkiJOPQDAjBdMQ8wDQYDVQQGEWZDYW5hZGExEDAOBgNVBAGMB09udGFyaW8xejAQ
BgNVBAsMCVNmhmR1bG1hbJEkMCIgA1UEAwbaG1naHdheS10ZXN0LmV4YW1wbGUuY29tIENBMCAX
DTIwMDIwMzA2NDcyMfoYDzI5OTkxMjMxMDAwMDAwWjAcMRowGAYDVQQQFDBEwMC1EMClFNS1GMi0w
MC0wMjBZMBMBGByqGSM49AgEGCCqGSM49AwEHA0IABAOjduOhs3zACpqHnK329DgTHNQMHb/gQE5B
efBbE0sgcbNEM0li8RYwzrsAfYcXp9k7E1CbpielT8OWf0z+ISejWTBxMB0GA1UdDgQWBBERFiMyW
lgBkn7C6I2VkZfQIBmxWrTAJBGNVHRMEAjaAMCsGCCsGAQUFBwEgBB8MHWhpZ2h3YXktZGVzdC5l
eGFtcGx1LmNvbTo5NDQzMAoGCCqGSM49BAMCA2gAMGUCMCPHqs7vIhI0WqXCFdYove091tbOBjXv
p8jcGKgx7gENPK3TXmKZyIkA0/FzdYGugLxALONXArQ/gSDkNNPbXKXsz4C6vHIWjJyWLDFA1B4
vAQdI14ib8N/jHzXm3AgkbThfzGCATswggE3AgEBMGUwXTEPMA0GA1UEBhMGQ2FuzYWRhMRawDgYD
VQQIDAdPbnRhcmlvMRiWEAYDVQQQLDAlTYW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktZGVzdC5l
eGFtcGx1LmNvbSBDQIEdYXcLTALBg1ghkgBZQMEAgGgaTAYBgkqhkiG9w0BQCMxYJKoZIhvcNAQcB
MBwGCSqGSIb3DQEJBTEPFw0yMDAyMjUyMzA0NDhaMC8GCSqGSIb3DQEJBDEiBCCx6IrwstHF
609Y0EqDK62QKby4duyyIWudvs15M16BBTAKBggqhkiJOPQDAgRHMEUCIBxwA1U1kIkuQDf/j7kZ
/MVefgr141+hKBFgrnNngjwpAIEAy8aXt0GSB9m1bmiEUpefCEhxSv2xLYurG1ugv0dfr/E=
<CODE ENDS>
```

The ASN1 decoding of the artifact:

```
file: examples/vr_00-D0-E5-F2-00-02.b64
```

```
0:d=0 hl=4 l=1759 cons: SEQUENCE
4:d=1 hl=2 l= 9 prim: OBJECT :pkcs7-signedData
15:d=1 hl=4 l=1744 cons: cont [ 0 ]
19:d=2 hl=4 l=1740 cons: SEQUENCE
23:d=3 hl=2 l= 1 prim: INTEGER :01
26:d=3 hl=2 l= 13 cons: SET
28:d=4 hl=2 l= 11 cons: SEQUENCE
30:d=5 hl=2 l= 9 prim: OBJECT :sha256
41:d=3 hl=4 l= 905 cons: SEQUENCE
45:d=4 hl=2 l= 9 prim: OBJECT :pkcs7-data
```

```

56:d=4  hl=4  l= 890 cons: cont [ 0 ]
60:d=5  hl=4  l= 886 prim: OCTET STRING      :{"ietf-voucher-request:v
950:d=3  hl=4  l= 490 cons: cont [ 0 ]
954:d=4  hl=4  l= 486 cons: SEQUENCE
958:d=5  hl=4  l= 364 cons: SEQUENCE
962:d=6  hl=2  l=   3 cons: cont [ 0 ]
964:d=7  hl=2  l=   1 prim: INTEGER           :02
967:d=6  hl=2  l=   4 prim: INTEGER           :0D85DC2D
973:d=6  hl=2  l=  10 cons: SEQUENCE
975:d=7  hl=2  l=   8 prim: OBJECT            :ecdsa-with-SHA256
985:d=6  hl=2  l=  93 cons: SEQUENCE
987:d=7  hl=2  l=  15 cons: SET
989:d=8  hl=2  l=  13 cons: SEQUENCE
991:d=9  hl=2  l=   3 prim: OBJECT            :countryName
996:d=9  hl=2  l=   6 prim: PRINTABLESTRING   :Canada
1004:d=7  hl=2  l=  16 cons: SET
1006:d=8  hl=2  l=  14 cons: SEQUENCE
1008:d=9  hl=2  l=   3 prim: OBJECT            :stateOrProvinceName
1013:d=9  hl=2  l=   7 prim: UTF8STRING       :Ontario
1022:d=7  hl=2  l=  18 cons: SET
1024:d=8  hl=2  l=  16 cons: SEQUENCE
1026:d=9  hl=2  l=   3 prim: OBJECT            :organizationalUnitName
1031:d=9  hl=2  l=   9 prim: UTF8STRING       :Sandelman
1042:d=7  hl=2  l=  36 cons: SET
1044:d=8  hl=2  l=  34 cons: SEQUENCE
1046:d=9  hl=2  l=   3 prim: OBJECT            :commonName
1051:d=9  hl=2  l=  27 prim: UTF8STRING       :highway-test.example.com
1080:d=6  hl=2  l=  32 cons: SEQUENCE
1082:d=7  hl=2  l=  13 prim: UTCTIME           :200203064720Z
1097:d=7  hl=2  l=  15 prim: GENERALIZEDTIME    :29991231000000Z
1114:d=6  hl=2  l=  28 cons: SEQUENCE
1116:d=7  hl=2  l=  26 cons: SET
1118:d=8  hl=2  l=  24 cons: SEQUENCE
1120:d=9  hl=2  l=   3 prim: OBJECT            :serialNumber
1125:d=9  hl=2  l=  17 prim: UTF8STRING       :00-D0-E5-F2-00-02
1144:d=6  hl=2  l=  89 cons: SEQUENCE
1146:d=7  hl=2  l=  19 cons: SEQUENCE
1148:d=8  hl=2  l=   7 prim: OBJECT            :id-ecPublicKey
1157:d=8  hl=2  l=   8 prim: OBJECT            :prime256v1
1167:d=7  hl=2  l=  66 prim: BIT STRING
1235:d=6  hl=2  l=  89 cons: cont [ 3 ]
1237:d=7  hl=2  l=  87 cons: SEQUENCE
1239:d=8  hl=2  l=  29 cons: SEQUENCE
1241:d=9  hl=2  l=   3 prim: OBJECT            :X509v3 Subject Key Ident
1246:d=9  hl=2  l=  22 prim: OCTET STRING      [HEX DUMP]:04144588CC9696
1270:d=8  hl=2  l=   9 cons: SEQUENCE
1272:d=9  hl=2  l=   3 prim: OBJECT            :X509v3 Basic Constraints
1277:d=9  hl=2  l=   2 prim: OCTET STRING      [HEX DUMP]:3000

```

```

1281:d=8 hl=2 l= 43 cons: SEQUENCE
1283:d=9 hl=2 l=  8 prim: OBJECT           :1.3.6.1.5.5.7.1.32
1293:d=9 hl=2 l= 31 prim: OCTET STRING     [HEX DUMP]:0C1D6869676877
1326:d=5 hl=2 l= 10 cons: SEQUENCE
1328:d=6 hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
1338:d=5 hl=2 l= 104 prim: BIT STRING
1444:d=3 hl=4 l= 315 cons: SET
1448:d=4 hl=4 l= 311 cons: SEQUENCE
1452:d=5 hl=2 l=  1 prim: INTEGER           :01
1455:d=5 hl=2 l= 101 cons: SEQUENCE
1457:d=6 hl=2 l= 93 cons: SEQUENCE
1459:d=7 hl=2 l= 15 cons: SET
1461:d=8 hl=2 l= 13 cons: SEQUENCE
1463:d=9 hl=2 l=  3 prim: OBJECT           :countryName
1468:d=9 hl=2 l=  6 prim: PRINTABLESTRING  :Canada
1476:d=7 hl=2 l= 16 cons: SET
1478:d=8 hl=2 l= 14 cons: SEQUENCE
1480:d=9 hl=2 l=  3 prim: OBJECT           :stateOrProvinceName
1485:d=9 hl=2 l=  7 prim: UTF8STRING       :Ontario
1494:d=7 hl=2 l= 18 cons: SET
1496:d=8 hl=2 l= 16 cons: SEQUENCE
1498:d=9 hl=2 l=  3 prim: OBJECT           :organizationalUnitName
1503:d=9 hl=2 l=  9 prim: UTF8STRING       :Sandelman
1514:d=7 hl=2 l= 36 cons: SET
1516:d=8 hl=2 l= 34 cons: SEQUENCE
1518:d=9 hl=2 l=  3 prim: OBJECT           :commonName
1523:d=9 hl=2 l= 27 prim: UTF8STRING       :highway-test.example.com
1552:d=6 hl=2 l=  4 prim: INTEGER           :0D85DC2D
1558:d=5 hl=2 l= 11 cons: SEQUENCE
1560:d=6 hl=2 l=  9 prim: OBJECT           :sha256
1571:d=5 hl=2 l= 105 cons: cont [ 0 ]
1573:d=6 hl=2 l= 24 cons: SEQUENCE
1575:d=7 hl=2 l=  9 prim: OBJECT           :contentType
1586:d=7 hl=2 l= 11 cons: SET
1588:d=8 hl=2 l=  9 prim: OBJECT           :pkcs7-data
1599:d=6 hl=2 l= 28 cons: SEQUENCE
1601:d=7 hl=2 l=  9 prim: OBJECT           :signingTime
1612:d=7 hl=2 l= 15 cons: SET
1614:d=8 hl=2 l= 13 prim: UTCTIME           :200225230448Z
1629:d=6 hl=2 l= 47 cons: SEQUENCE
1631:d=7 hl=2 l=  9 prim: OBJECT           :messageDigest
1642:d=7 hl=2 l= 34 cons: SET
1644:d=8 hl=2 l= 32 prim: OCTET STRING     [HEX DUMP]:B1E88AF0B2D1C5
1678:d=5 hl=2 l= 10 cons: SEQUENCE
1680:d=6 hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
1690:d=5 hl=2 l= 71 prim: OCTET STRING     [HEX DUMP]:304502201C7003

```

The JSON contained in the voucher request:

U2Eyb3ZXRFJEVfZGRE9ISk5Ua0p6VEc5T2NqRk1OVzVITlRabWQwRmtTVGhvYVVGWFJ6aFRPRmhC
VWpWck1Vtm5lRE5aV1ZGQ1UyZGtVMk5HWTBGa1ppc3JRbmMyV1hrc1ZUMGlmWDJnZ2dIcU1JSUI1
akNDQVd5Z0F3SUJBZ0lFRFlYY0xUQUtCZ2dxaGtqT1BRUURBakJkTVE4d0RRWURWUVFHRXdaRF1X
NWHaR0V4RURBT0JnTlZCQWdNQjA5dWRHRn1hVzh4RWpBUUJnTlZCQXNNQ1ZOaGJtUmxiRzFoYmpF
a01DSUdBMVVFQXd3YmFhbG5hSGRoZVMxMFPyTjBmbyV0WVcxZ2JHVXVZMj10SUVOQk1DQVhEVEl3
TURJd016QTJORG5TUZvWUR6STVPVgt4TWpNeE1EQXdNREF3V2pBY01Sb3dHQV1EV1FRrkRCRXdN
QzFFtUMxRk5TMudNaTB3TUMwd01qQ1pNqk1HqnlxR1NNND1BZ0VHQ0Nxr1NNND1Bd0VIQTBjQUJB
T2pkVU9IczN6QUNwcUhuSzMyOURnVEhOUUoQ19nUUU1QmVmQmJFMHnY2JORW0wbGk4U1L3enJz
QWZZQ3hwOWs3RTFDYnBpZwXUOE9XZjB6K01TZWpXVEJYTU1wR0ExVWREZ1FXQkJSRmlNeVdsZ0Jr
TjdDnkkyVmtaRlFJQm14V3JUQUpCZ05WSFJNRUFqQUFNQ3NHQ0NzR0FRVUZCd0VnQkI4TUhXaHBA
MmgzWVhrdGRHVnpkQzVsZUdGdGNHeGxMbU52Y1RvNU5EUXpNQW9HQ0Nxr1NNND1CQU1DQJnQU1H
VUNNQ1BocVM3dk1oSTBXcVhDRmRZb3ZlMD1sdGJPQkpYdnA4amNHS2d4eDdnRU5QSzNUWG1LWnlJ
a0EwL0Z6ZF1HdWdJeEFMT05YXJRL2dTRGtOT1BiWEtYc3o0QzZ2SElXakp5V0xkRkFsQjR2QVfK
STE0aWI4T19qSHpYbTnBZ2tiVGhmeKdQVRzZ2dnRtNBZ0VCTUdVd1hURVBNQTBHQTFVFRUJ0TUdR
MkZ1WVdSaE1SQXdEZ11EV1FRSURBZFBib1JoY21sdK1SSXdFQV1EV1FRTERBbFRZVzVrW1d4dFlX
NHhKREFPqmdOVkBTU1HMmhWwjJoM1lYa3RkR1Z6ZEM1bGVHRnRjR3hsTG1OdmJTQkRRUU1FRFlY
Y0xUQUxZ2xnaGtnQ1pRTUUVBZ0dnYVRBWUJna3Foa2lHOXcwQkNRTXhdD1lKS29aSWH2Y05BUWNC
TUJ3R0NTcUdTSWIZRFFFskJURVbGdzB5TURBeU1qVX1NekEwTkRoYU1DOEdDU3FHU0liMORRRUpC
REVPqKNDedZJcndzdEhGNjA5WTBfCURLNjJRS2J5NGR1eXlJV3VkdNMXNU0xNkJCVEFLQmdncWhr
ak9QUVFEQWdSSE1FVUNJQnh3QTFVbGtJa3VRRGYvaJdrWi9NVmVmZ3IxNDEraEtCRmdybk5uZ2p3
cEFpRUF50GFYdDBHU0I5bTFibW1FVXB1ZkNFaHhTdjJ4TF1lckdsdWd2MGRmci9FPJS9faCCBG8w
ggH8MIIBGqADAgECAgQ/mJtSMAoGCCqGSM49BAMCMG0xeJAQBgOJkiaJk/IsZAEZFgJjYTEZMBcG
CgmSjomT8ixkARKWCXNhbMR1bG1hbje8MDoGA1UEAwZm91bnRhaW4tdGVzdC5leGftcGx1LmNv
bSBVbnN0cnVuZyBGB3VudGFpbiBSb290IENBMB4XDTEwMDIyNTIxMzE1NFoXDTEyMDIyNDIxMzE1
NFowUzESMBAGCgmSjomT8ixkARKWAmNhMRkWFwYKcZImiZPyLgQBGRYJc2FuzGVsbWVfMTwwOgYD
VQDDb1mb3VudGFpbi10ZXN0LmV4YW1wbGUuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE
lmVQcjs6n+Xd5l/28IFv6UiegQwSBztGj5dkK2MAjQIPV8181H+EjLlIOYdbJiI0VtEiF1/Jqt+TO
BfinTNOLOqMqMCgWfGyDVR0LAQH/BAwwCgYIKwYBBQUHAXwDgYDVR0PAQH/BAQDAgeAMAoGCCqG
SM49BAMCA2gAMGUCMGPYExVSB6WB/jdH7nIEo1FNoeBI8C8u/HLPsYVVM9fH7/VHA5qCa8bdpeZ
GSP9fgIXALysw0Gwug2vUvmcbnp/AB0jyGIBYbxLxcBHmTUKDHdhRAFKB1JwVwB1/74HDpjL5TCC
AmswggHyOAMCAQICBC1rBlkwCgYIKoZIzj0EAwIwbTESMBAGCgmSjomT8ixkARKWAmNhMRkWFwYK
CZImiZPyLgQBGRYJc2FuzGVsbWVfMTwwOgYDQVQDDNmb3VudGFpbi10ZXN0LmV4YW1wbGUuY29t
IFVuc3Rydw5nIEZvdW50YW1uIFJvb3QgQ0EwHhcNMjAwMjI1MjEzMTQ1WhcNMjAwMjI1MjEzMTQ1
WjBtMRIwEAYKcZImiZPyLgQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4xPDA6BgNV
BAMMM2ZvdW50YW1uLXRlc3QuZlZhbXhBXS55j20gVW5zdHJ1bmcgRm91bnRhaW4gUm9vdCBDQTB2
MBAGByqGSM49AgEGBSuBBAAiA2IABBT/WboXwxq8Zo2MbODD+jfxD2X2Ipg9t1aAB9vfuHq1RU15
ikaXGVMWmBGPax0yvjzIPltjtUb2qNVvm/nA8905FD9yR1Gkdt3S8L/1yo8wAX/4w1/T9SADRIuL
8gdstKNjMGEwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFml19ssR
4QekSSynCMZ8ELyHs3QmMB8GA1UdIwQYMBaAFml19ssR4QekSSynCMZ8ELyHs3QmMAoGCCqGSM49
BAMCA2cAMGQCMCCDBs6NmKRUMZMSjppw1I2W1KNWX0gmyppFFiHONhVed39KtiVHpGTdrT1Zi1E
tAIwbzj5rxLtMNWFKXyXfli9Z5FDxA0w+dgcrC8G3bzVBkIshKIE6gKkXxdRjvvZL9JcMYIBSzcC
AUcCAQEwdTBTMRIwEAYKcZImiZPyLgQBGRYCY2ExGTAXBgoJkiaJk/IsZAEZFglzYW5kZWxtYW4x
PDA6BgNVBAMMM2ZvdW50YW1uLXRlc3QuZlZhbXhBXS55j20gVW5zdHJ1bmcgRm91bnRhaW4gUm9v
dCBDQQIEP5ibUjALBg1ghkgBZQMEAgGgATAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcCBMwGCCqG
SIB3DQEJBTepFw0yMDAyMjUyMzA0NDlaMCC8GCSqGSIB3DQEJBDiBCA9gYXR1sS0giII3PwvOK/N
5RUBwjsL/cDcrH/Bd+E1ajAKBggqhkiJOPQDdAgRHMEUCIFieXZa07P9eZMpCvN2laB4czw7I0s0P
s9+frcJtEBTTAiEAhCcB//qmgqcEA+90mquvVNENmFH9dxCH8Ihhz6SCVDI=
<CODE ENDS>

The ASN1 decoding of the artifact:

file: examples/parboiled_vr_00_D0-E5-02-00-2D.b64

```

0:d=0 hl=4 l=4087 cons: SEQUENCE
4:d=1 hl=2 l= 9 prim: OBJECT           :pkcs7-signedData
15:d=1 hl=4 l=4072 cons: cont [ 0 ]
19:d=2 hl=4 l=4068 cons: SEQUENCE
23:d=3 hl=2 l= 1 prim: INTEGER         :01
26:d=3 hl=2 l= 13 cons: SET
28:d=4 hl=2 l= 11 cons: SEQUENCE
30:d=5 hl=2 l= 9 prim: OBJECT          :sha256
41:d=3 hl=4 l=2572 cons: SEQUENCE
45:d=4 hl=2 l= 9 prim: OBJECT          :pkcs7-data
56:d=4 hl=4 l=2557 cons: cont [ 0 ]
60:d=5 hl=4 l=2553 prim: OCTET STRING  :{"ietf-voucher-request:v
2617:d=3 hl=4 l=1135 cons: cont [ 0 ]
2621:d=4 hl=4 l= 508 cons: SEQUENCE
2625:d=5 hl=4 l= 386 cons: SEQUENCE
2629:d=6 hl=2 l= 3 cons: cont [ 0 ]
2631:d=7 hl=2 l= 1 prim: INTEGER       :02
2634:d=6 hl=2 l= 4 prim: INTEGER       :3F989B52
2640:d=6 hl=2 l= 10 cons: SEQUENCE
2642:d=7 hl=2 l= 8 prim: OBJECT        :ecdsa-with-SHA256
2652:d=6 hl=2 l= 109 cons: SEQUENCE
2654:d=7 hl=2 l= 18 cons: SET
2656:d=8 hl=2 l= 16 cons: SEQUENCE
2658:d=9 hl=2 l= 10 prim: OBJECT       :domainComponent
2670:d=9 hl=2 l= 2 prim: IA5STRING     :ca
2674:d=7 hl=2 l= 25 cons: SET
2676:d=8 hl=2 l= 23 cons: SEQUENCE
2678:d=9 hl=2 l= 10 prim: OBJECT       :domainComponent
2690:d=9 hl=2 l= 9 prim: IA5STRING     :sandelman
2701:d=7 hl=2 l= 60 cons: SET
2703:d=8 hl=2 l= 58 cons: SEQUENCE
2705:d=9 hl=2 l= 3 prim: OBJECT        :commonName
2710:d=9 hl=2 l= 51 prim: UTF8STRING   :fountain-test.example.co
2763:d=6 hl=2 l= 30 cons: SEQUENCE
2765:d=7 hl=2 l= 13 prim: UTCTIME     :200225213154Z
2780:d=7 hl=2 l= 13 prim: UTCTIME     :220224213154Z
2795:d=6 hl=2 l= 83 cons: SEQUENCE
2797:d=7 hl=2 l= 18 cons: SET
2799:d=8 hl=2 l= 16 cons: SEQUENCE
2801:d=9 hl=2 l= 10 prim: OBJECT       :domainComponent
2813:d=9 hl=2 l= 2 prim: IA5STRING     :ca
2817:d=7 hl=2 l= 25 cons: SET
2819:d=8 hl=2 l= 23 cons: SEQUENCE
2821:d=9 hl=2 l= 10 prim: OBJECT       :domainComponent

```

```

2833:d=9 hl=2 l= 9 prim: IA5STRING :sandelman
2844:d=7 hl=2 l= 34 cons: SET
2846:d=8 hl=2 l= 32 cons: SEQUENCE
2848:d=9 hl=2 l= 3 prim: OBJECT :commonName
2853:d=9 hl=2 l= 25 prim: UTF8STRING :fountain-test.example.co
2880:d=6 hl=2 l= 89 cons: SEQUENCE
2882:d=7 hl=2 l= 19 cons: SEQUENCE
2884:d=8 hl=2 l= 7 prim: OBJECT :id-ecPublicKey
2893:d=8 hl=2 l= 8 prim: OBJECT :prime256v1
2903:d=7 hl=2 l= 66 prim: BIT STRING
2971:d=6 hl=2 l= 42 cons: cont [ 3 ]
2973:d=7 hl=2 l= 40 cons: SEQUENCE
2975:d=8 hl=2 l= 22 cons: SEQUENCE
2977:d=9 hl=2 l= 3 prim: OBJECT :X509v3 Extended Key Usag
2982:d=9 hl=2 l= 1 prim: BOOLEAN :255
2985:d=9 hl=2 l= 12 prim: OCTET STRING [HEX DUMP]:300A06082B0601
2999:d=8 hl=2 l= 14 cons: SEQUENCE
3001:d=9 hl=2 l= 3 prim: OBJECT :X509v3 Key Usage
3006:d=9 hl=2 l= 1 prim: BOOLEAN :255
3009:d=9 hl=2 l= 4 prim: OCTET STRING [HEX DUMP]:03020780
3015:d=5 hl=2 l= 10 cons: SEQUENCE
3017:d=6 hl=2 l= 8 prim: OBJECT :ecdsa-with-SHA256
3027:d=5 hl=2 l= 104 prim: BIT STRING
3133:d=4 hl=4 l= 619 cons: SEQUENCE
3137:d=5 hl=4 l= 498 cons: SEQUENCE
3141:d=6 hl=2 l= 3 cons: cont [ 0 ]
3143:d=7 hl=2 l= 1 prim: INTEGER :02
3146:d=6 hl=2 l= 4 prim: INTEGER :296B0659
3152:d=6 hl=2 l= 10 cons: SEQUENCE
3154:d=7 hl=2 l= 8 prim: OBJECT :ecdsa-with-SHA256
3164:d=6 hl=2 l= 109 cons: SEQUENCE
3166:d=7 hl=2 l= 18 cons: SET
3168:d=8 hl=2 l= 16 cons: SEQUENCE
3170:d=9 hl=2 l= 10 prim: OBJECT :domainComponent
3182:d=9 hl=2 l= 2 prim: IA5STRING :ca
3186:d=7 hl=2 l= 25 cons: SET
3188:d=8 hl=2 l= 23 cons: SEQUENCE
3190:d=9 hl=2 l= 10 prim: OBJECT :domainComponent
3202:d=9 hl=2 l= 9 prim: IA5STRING :sandelman
3213:d=7 hl=2 l= 60 cons: SET
3215:d=8 hl=2 l= 58 cons: SEQUENCE
3217:d=9 hl=2 l= 3 prim: OBJECT :commonName
3222:d=9 hl=2 l= 51 prim: UTF8STRING :fountain-test.example.co
3275:d=6 hl=2 l= 30 cons: SEQUENCE
3277:d=7 hl=2 l= 13 prim: UTCTIME :200225213145Z
3292:d=7 hl=2 l= 13 prim: UTCTIME :220224213145Z
3307:d=6 hl=2 l= 109 cons: SEQUENCE
3309:d=7 hl=2 l= 18 cons: SET

```

```

3311:d=8  hl=2 l= 16 cons: SEQUENCE
3313:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3325:d=9  hl=2 l=  2 prim: IA5STRING        :ca
3329:d=7  hl=2 l= 25 cons: SET
3331:d=8  hl=2 l= 23 cons: SEQUENCE
3333:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3345:d=9  hl=2 l=  9 prim: IA5STRING        :sandelman
3356:d=7  hl=2 l= 60 cons: SET
3358:d=8  hl=2 l= 58 cons: SEQUENCE
3360:d=9  hl=2 l=  3 prim: OBJECT           :commonName
3365:d=9  hl=2 l= 51 prim: UTF8STRING       :fountain-test.example.co
3418:d=6  hl=2 l= 118 cons: SEQUENCE
3420:d=7  hl=2 l= 16 cons: SEQUENCE
3422:d=8  hl=2 l=  7 prim: OBJECT           :id-ecPublicKey
3431:d=8  hl=2 l=  5 prim: OBJECT           :secp384r1
3438:d=7  hl=2 l= 98 prim: BIT STRING
3538:d=6  hl=2 l= 99 cons: cont [ 3 ]
3540:d=7  hl=2 l= 97 cons: SEQUENCE
3542:d=8  hl=2 l= 15 cons: SEQUENCE
3544:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Basic Constraints
3549:d=9  hl=2 l=  1 prim: BOOLEAN            :255
3552:d=9  hl=2 l=  5 prim: OCTET STRING      [HEX DUMP]:30030101FF
3559:d=8  hl=2 l= 14 cons: SEQUENCE
3561:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Key Usage
3566:d=9  hl=2 l=  1 prim: BOOLEAN            :255
3569:d=9  hl=2 l=  4 prim: OCTET STRING      [HEX DUMP]:03020106
3575:d=8  hl=2 l= 29 cons: SEQUENCE
3577:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Subject Key Ident
3582:d=9  hl=2 l= 22 prim: OCTET STRING      [HEX DUMP]:0414B9A5F6CB11
3606:d=8  hl=2 l= 31 cons: SEQUENCE
3608:d=9  hl=2 l=  3 prim: OBJECT           :X509v3 Authority Key Ide
3613:d=9  hl=2 l= 24 prim: OCTET STRING      [HEX DUMP]:30168014B9A5F6
3639:d=5  hl=2 l= 10 cons: SEQUENCE
3641:d=6  hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
3651:d=5  hl=2 l= 103 prim: BIT STRING
3756:d=3  hl=4 l= 331 cons: SET
3760:d=4  hl=4 l= 327 cons: SEQUENCE
3764:d=5  hl=2 l=  1 prim: INTEGER            :01
3767:d=5  hl=2 l= 117 cons: SEQUENCE
3769:d=6  hl=2 l= 109 cons: SEQUENCE
3771:d=7  hl=2 l= 18 cons: SET
3773:d=8  hl=2 l= 16 cons: SEQUENCE
3775:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3787:d=9  hl=2 l=  2 prim: IA5STRING        :ca
3791:d=7  hl=2 l= 25 cons: SET
3793:d=8  hl=2 l= 23 cons: SEQUENCE
3795:d=9  hl=2 l= 10 prim: OBJECT           :domainComponent
3807:d=9  hl=2 l=  9 prim: IA5STRING        :sandelman

```

```
3818:d=7 hl=2 l= 60 cons: SET
3820:d=8 hl=2 l= 58 cons: SEQUENCE
3822:d=9 hl=2 l=  3 prim: OBJECT           :commonName
3827:d=9 hl=2 l= 51 prim: UTF8STRING      :fountain-test.example.co
3880:d=6 hl=2 l=  4 prim: INTEGER         :3F989B52
3886:d=5 hl=2 l= 11 cons: SEQUENCE
3888:d=6 hl=2 l=  9 prim: OBJECT           :sha256
3899:d=5 hl=2 l= 105 cons: cont [ 0 ]
3901:d=6 hl=2 l= 24 cons: SEQUENCE
3903:d=7 hl=2 l=  9 prim: OBJECT           :contentType
3914:d=7 hl=2 l= 11 cons: SET
3916:d=8 hl=2 l=  9 prim: OBJECT           :pkcs7-data
3927:d=6 hl=2 l= 28 cons: SEQUENCE
3929:d=7 hl=2 l=  9 prim: OBJECT           :signingTime
3940:d=7 hl=2 l= 15 cons: SET
3942:d=8 hl=2 l= 13 prim: UTCTIME         :200225230449Z
3957:d=6 hl=2 l= 47 cons: SEQUENCE
3959:d=7 hl=2 l=  9 prim: OBJECT           :messageDigest
3970:d=7 hl=2 l= 34 cons: SET
3972:d=8 hl=2 l= 32 prim: OCTET STRING    [HEX DUMP]:3D818C51D6C4B4
4006:d=5 hl=2 l= 10 cons: SEQUENCE
4008:d=6 hl=2 l=  8 prim: OBJECT           :ecdsa-with-SHA256
4018:d=5 hl=2 l= 71 prim: OCTET STRING    [HEX DUMP]:30450220589E5D
```

The JSON contained in the voucher request. Note that the previous voucher request is in the prior-signed-voucher-request attribute.


```
<CODE BEGINS> file "voucher_00-D0-E5-F2-00-02.b64"
MIIGxwYJKoZIhvcNAQcCoIIGuDCCBRQCAQEExDTALBg1ghkgBZQMEAgEwggn4BgkqhkiG9w0BBWgg
ggNpBIIDZXsiaWV0Zi12b3VjaGVyOnZvdWNoZXIiOansiYXNzZXJ0aW9uIjoibG9nZ2VkIiwia3Jl
YXRlZC1vbiI6IjIwMjAtMDItMjVUMTg6MDQ6NDkuMzAzLTA1OjAwIiwic2VyaWFnLW51bWJlciI6
IjAwLUUwLUU1LUYyLTAwLTAYIiwibm9uY2UiOiJhTWpndWVlVWQtMjJ3VmltajZ6MjdRIiwicGlu
bmVkbWVkbWVpbi1jZXJ0IjoitU1JQi9EQ0NBWUtncXJkQkFnSUVQNWliVWpBS0JnZ3Foa2pPUFFR
REFqQnRNUk13RUFZS0NaSW1pW1B5TEdRQkdSWUNZMkV4R1RBWEJnb0praWFfKay9Jc1pBRVpGZ2x6
WVc1a1pXeHRZVzR4UERBNk1JnTlZCQU1NTTJaadmRXNTBZV2x1TFhSbGMzUXVaWGhoYlhCc1pTNWpi
MjBnV1c1emRISjFibWNNUm05MWJuUmhhVzRnVW05dmRDQkRRVEFlRncweU1EQX1Na1V5TVRNeE5U
UmFGdzB5TWpBeU1qUX1NVE14TlRSYU1GTXhFakFRQmdvSmtYUprL0lZwKFFWkZnSmpZVEVaTUJj
R0NnbVNBk21UOgl4a0fSa1dDWE5oYm1SbGJHMWhiakVpTUNBR0ExVUVBd3daWm05MWJuUmhhVzR0
ZEdWemRDNWx1R0Z0Y0d4bExtTnZiVEJaTUJNR0J5cUdTTTQ5QWdFR0NDcUdTTTQ5QXdfSEEWsUFC
SlpsVUhJMhVwL2wzZVpmOXZDQmIrbElub0VNRWdjN1JvK1haQ3RqQUkwQ0xZkpmSlIvaE15eURt
SFd5VWlORmJSQ0g5Zn1hcmZremdYNHAWelRpenFqS2pBb01CWUdBMVVKs1LFFQi93U1NQW9HQONz
R0FRVUZCd01jTUE0R0ExVWREd0VCL3dRRUF3SUhnREFLQmdncWhrak9QUVFEQWd0b0FEQmxBakJt
VDJCTVZVZ2VsZ2Y0M1IrNX1CS05SVGFibX1lQXZMdnh5ejBtRlZad1h4Ky8xUndPYWdtckczYVht
UmtqL1g0Q01RQzhyTU5Cc0xvTnIxTDVurZU2ZndBZEK4aGlBV0c4UzhYQVI1azFDZ3gzWVVRQ1Nn
ZFNjRmNBZGYrK0J3N115K1U9In19oIIB4zCCAd8wggFkoAMCAQICBBuZX1QwCgYIKoZIzj0EAwIw
XTEPMA0GA1UEBhMGQ2FuwYWRhMRAwDgYDZDQkIDAdPbnRhcm1vMRIwEAYDVQQDLDA1TYW5kZWxtYW4x
JDAiBgNVBAMMG2hpZ2h3YXktZGVzZDc5leGFtcGxlLmNvbSBBDQTAeFw0xOTAyMTIyMjIyNDFAFw0y
MTAyMTEyMjIyNDFAFw0yMTAyMTEyMjIyNDFAFw0yMTAyMTEyMjIyNDFAFw0yMTAyMTEyMjIyNDFA
CwwJU2FuzGVsbWVfUmsYwJAYDVQQDDB1oaWdod2F5LXRlc3QuZXhhbXBsZS5jb20gTUFTQTZBZMBMG
ByqGSM49AgEGCCqGSM49AwEHA0IABK0EfaNEueJE+Mn5GwcbpnRznB66bKmqTCpojjZ96AdRwFt
uTCVfoKouLTBX0idIhMLfJLM31lyuKy4CUtpp6WjEDAOMAwGA1UdEwEB/wQMAAwCgYIKoZIzj0EA
AwIDAQAwZgIxAL1V5ZsO+/xelSnjgbMVNaqTKIEvkrYs1F9TW3r0dXBEDqyOxtXP8XMsKMO55lG
ugIxAPZ/RH23FPrrRZ2rUEcNLRub7mphW+oUhlLxITPA/8ps/roggp675cv9b+XhozW9IyTGCAIsw
ggE3AgEBMGUwXTEPMA0GA1UEBhMGQ2FuwYWRhMRAwDgYDZDQkIDAdPbnRhcm1vMRIwEAYDVQQDLDA1T
YW5kZWxtYW4xJDAiBgNVBAMMG2hpZ2h3YXktZGVzZDc5leGFtcGxlLmNvbSBBDQQIEG51fVDALBg1g
hkgBZQMEAgGgaTAYBgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0yMDAy
MjUyMzA0NDlaMC8GCSqGSIb3DQEJBDEiBCCJQso4Z9msdaPk3bsDltTkVckX50DvOPuOR9Svi5M9
RDAKBggqhkiJOPQDAGRHMEUCIQCKESXfM3iV8hpkqcxAKA1veArA6GFpN0jzys4E18uDgIgsRQi
9/MntuJhAM/tJCZBkfHBoAGX4PFAwwbs5LFZtAw=
<CODE ENDS>
```

The ASN1 decoding of the artifact:

```
file: examples/voucher_00-D0-E5-F2-00-02.b64
```

```
0:d=0  hl=4  l=1735  cons: SEQUENCE
4:d=1  hl=2  l=   9  prim: OBJECT                :pkcs7-signedData
15:d=1  hl=4  l=1720  cons: cont [ 0 ]
19:d=2  hl=4  l=1716  cons: SEQUENCE
23:d=3  hl=2  l=   1  prim: INTEGER                :01
26:d=3  hl=2  l=  13  cons: SET
28:d=4  hl=2  l=  11  cons: SEQUENCE
30:d=5  hl=2  l=   9  prim: OBJECT                :sha256
41:d=3  hl=4  l= 888  cons: SEQUENCE
45:d=4  hl=2  l=   9  prim: OBJECT                :pkcs7-data
```

```

56:d=4  hl=4  l= 873 cons: cont [ 0 ]
60:d=5  hl=4  l= 869 prim: OCTET STRING      :{"ietf-voucher:voucher":
933:d=3  hl=4  l= 483 cons: cont [ 0 ]
937:d=4  hl=4  l= 479 cons: SEQUENCE
941:d=5  hl=4  l= 356 cons: SEQUENCE
945:d=6  hl=2  l=   3 cons: cont [ 0 ]
947:d=7  hl=2  l=   1 prim: INTEGER           :02
950:d=6  hl=2  l=   4 prim: INTEGER           :1B995F54
956:d=6  hl=2  l=  10 cons: SEQUENCE
958:d=7  hl=2  l=   8 prim: OBJECT            :ecdsa-with-SHA256
968:d=6  hl=2  l=  93 cons: SEQUENCE
970:d=7  hl=2  l=  15 cons: SET
972:d=8  hl=2  l=  13 cons: SEQUENCE
974:d=9  hl=2  l=   3 prim: OBJECT            :countryName
979:d=9  hl=2  l=   6 prim: PRINTABLESTRING   :Canada
987:d=7  hl=2  l=  16 cons: SET
989:d=8  hl=2  l=  14 cons: SEQUENCE
991:d=9  hl=2  l=   3 prim: OBJECT            :stateOrProvinceName
996:d=9  hl=2  l=   7 prim: UTF8STRING        :Ontario
1005:d=7  hl=2  l=  18 cons: SET
1007:d=8  hl=2  l=  16 cons: SEQUENCE
1009:d=9  hl=2  l=   3 prim: OBJECT            :organizationalUnitName
1014:d=9  hl=2  l=   9 prim: UTF8STRING        :Sandelman
1025:d=7  hl=2  l=  36 cons: SET
1027:d=8  hl=2  l=  34 cons: SEQUENCE
1029:d=9  hl=2  l=   3 prim: OBJECT            :commonName
1034:d=9  hl=2  l=  27 prim: UTF8STRING        :highway-test.example.com
1063:d=6  hl=2  l=  30 cons: SEQUENCE
1065:d=7  hl=2  l=  13 prim: UTCTIME           :190212222241Z
1080:d=7  hl=2  l=  13 prim: UTCTIME           :210211222241Z
1095:d=6  hl=2  l=  95 cons: SEQUENCE
1097:d=7  hl=2  l=  15 cons: SET
1099:d=8  hl=2  l=  13 cons: SEQUENCE
1101:d=9  hl=2  l=   3 prim: OBJECT            :countryName
1106:d=9  hl=2  l=   6 prim: PRINTABLESTRING   :Canada
1114:d=7  hl=2  l=  16 cons: SET
1116:d=8  hl=2  l=  14 cons: SEQUENCE
1118:d=9  hl=2  l=   3 prim: OBJECT            :stateOrProvinceName
1123:d=9  hl=2  l=   7 prim: UTF8STRING        :Ontario
1132:d=7  hl=2  l=  18 cons: SET
1134:d=8  hl=2  l=  16 cons: SEQUENCE
1136:d=9  hl=2  l=   3 prim: OBJECT            :organizationalUnitName
1141:d=9  hl=2  l=   9 prim: UTF8STRING        :Sandelman
1152:d=7  hl=2  l=  38 cons: SET
1154:d=8  hl=2  l=  36 cons: SEQUENCE
1156:d=9  hl=2  l=   3 prim: OBJECT            :commonName
1161:d=9  hl=2  l=  29 prim: UTF8STRING        :highway-test.example.com
1192:d=6  hl=2  l=  89 cons: SEQUENCE

```

```

1194:d=7 hl=2 l= 19 cons: SEQUENCE
1196:d=8 hl=2 l= 7 prim: OBJECT           :id-ecPublicKey
1205:d=8 hl=2 l= 8 prim: OBJECT           :prime256v1
1215:d=7 hl=2 l= 66 prim: BIT STRING
1283:d=6 hl=2 l= 16 cons: cont [ 3 ]
1285:d=7 hl=2 l= 14 cons: SEQUENCE
1287:d=8 hl=2 l= 12 cons: SEQUENCE
1289:d=9 hl=2 l= 3 prim: OBJECT           :X509v3 Basic Constraints
1294:d=9 hl=2 l= 1 prim: BOOLEAN         :255
1297:d=9 hl=2 l= 2 prim: OCTET STRING    [HEX DUMP]:3000
1301:d=5 hl=2 l= 10 cons: SEQUENCE
1303:d=6 hl=2 l= 8 prim: OBJECT           :ecdsa-with-SHA256
1313:d=5 hl=2 l= 105 prim: BIT STRING
1420:d=3 hl=4 l= 315 cons: SET
1424:d=4 hl=4 l= 311 cons: SEQUENCE
1428:d=5 hl=2 l= 1 prim: INTEGER         :01
1431:d=5 hl=2 l= 101 cons: SEQUENCE
1433:d=6 hl=2 l= 93 cons: SEQUENCE
1435:d=7 hl=2 l= 15 cons: SET
1437:d=8 hl=2 l= 13 cons: SEQUENCE
1439:d=9 hl=2 l= 3 prim: OBJECT           :countryName
1444:d=9 hl=2 l= 6 prim: PRINTABLESTRING :Canada
1452:d=7 hl=2 l= 16 cons: SET
1454:d=8 hl=2 l= 14 cons: SEQUENCE
1456:d=9 hl=2 l= 3 prim: OBJECT           :stateOrProvinceName
1461:d=9 hl=2 l= 7 prim: UTF8STRING      :Ontario
1470:d=7 hl=2 l= 18 cons: SET
1472:d=8 hl=2 l= 16 cons: SEQUENCE
1474:d=9 hl=2 l= 3 prim: OBJECT           :organizationalUnitName
1479:d=9 hl=2 l= 9 prim: UTF8STRING      :Sandelman
1490:d=7 hl=2 l= 36 cons: SET
1492:d=8 hl=2 l= 34 cons: SEQUENCE
1494:d=9 hl=2 l= 3 prim: OBJECT           :commonName
1499:d=9 hl=2 l= 27 prim: UTF8STRING      :highway-test.example.com
1528:d=6 hl=2 l= 4 prim: INTEGER         :1B995F54
1534:d=5 hl=2 l= 11 cons: SEQUENCE
1536:d=6 hl=2 l= 9 prim: OBJECT           :sha256
1547:d=5 hl=2 l= 105 cons: cont [ 0 ]
1549:d=6 hl=2 l= 24 cons: SEQUENCE
1551:d=7 hl=2 l= 9 prim: OBJECT           :contentType
1562:d=7 hl=2 l= 11 cons: SET
1564:d=8 hl=2 l= 9 prim: OBJECT           :pkcs7-data
1575:d=6 hl=2 l= 28 cons: SEQUENCE
1577:d=7 hl=2 l= 9 prim: OBJECT           :signingTime
1588:d=7 hl=2 l= 15 cons: SET
1590:d=8 hl=2 l= 13 prim: UTCTIME         :200225230449Z
1605:d=6 hl=2 l= 47 cons: SEQUENCE
1607:d=7 hl=2 l= 9 prim: OBJECT           :messageDigest

```

```
1618:d=7 hl=2 l= 34 cons: SET
1620:d=8 hl=2 l= 32 prim: OCTET STRING [HEX DUMP]:8942CA3867D9AC
1654:d=5 hl=2 l= 10 cons: SEQUENCE
1656:d=6 hl=2 l= 8 prim: OBJECT :ecdsa-with-SHA256
1666:d=5 hl=2 l= 71 prim: OCTET STRING [HEX DUMP]:30450221008A11
```

Appendix D. Additional References

RFC EDITOR Please remove this section before publication. It exists just to include references to the things in the YANG descriptions which are not otherwise referenced in the text so that xml2rfc will not complain.

[ITU.X690.1994]

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael C. Richardson
Sandelman Software Works

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

Toerless Eckert
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara, CA 95050
United States of America

Email: tte+iETF@cs.fau.de

Michael H. Behringer

Email: Michael.H.Behringer@gmail.com

Kent Watsen
Watsen Networks

Email: kent+iETF@watsen.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2018

C. Bormann
Universitaet Bremen TZI
B. Carpenter, Ed.
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

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

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

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

This Internet-Draft will expire on January 8, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1.	Introduction	3
2.	GRASP Protocol Overview	5
2.1.	Terminology	5
2.2.	High Level Deployment Model	7
2.3.	High Level Design	8
2.4.	Quick Operating Overview	11
2.5.	GRASP Protocol Basic Properties and Mechanisms	12
2.5.1.	Required External Security Mechanism	12
2.5.2.	Discovery Unsolicited Link-Local (DULL) GRASP	13
2.5.3.	Transport Layer Usage	14
2.5.4.	Discovery Mechanism and Procedures	15
2.5.5.	Negotiation Procedures	19
2.5.6.	Synchronization and Flooding Procedures	21
2.6.	GRASP Constants	23
2.7.	Session Identifier (Session ID)	24
2.8.	GRASP Messages	25
2.8.1.	Message Overview	25
2.8.2.	GRASP Message Format	25
2.8.3.	Message Size	26
2.8.4.	Discovery Message	26
2.8.5.	Discovery Response Message	28
2.8.6.	Request Messages	29
2.8.7.	Negotiation Message	30
2.8.8.	Negotiation End Message	30
2.8.9.	Confirm Waiting Message	30
2.8.10.	Synchronization Message	31
2.8.11.	Flood Synchronization Message	31
2.8.12.	Invalid Message	32
2.8.13.	No Operation Message	33
2.9.	GRASP Options	33
2.9.1.	Format of GRASP Options	33
2.9.2.	Divert Option	33
2.9.3.	Accept Option	34
2.9.4.	Decline Option	34
2.9.5.	Locator Options	34
2.10.	Objective Options	36
2.10.1.	Format of Objective Options	36
2.10.2.	Objective flags	38

2.10.3.	General Considerations for Objective Options	38
2.10.4.	Organizing of Objective Options	39
2.10.5.	Experimental and Example Objective Options	41
3.	Implementation Status [RFC Editor: please remove]	41
3.1.	BUPT C++ Implementation	41
3.2.	Python Implementation	42
4.	Security Considerations	42
5.	CDDL Specification of GRASP	45
6.	IANA Considerations	47
7.	Acknowledgements	49
8.	References	49
8.1.	Normative References	49
8.2.	Informative References	50
Appendix A.	Open Issues [RFC Editor: This section should be empty. Please remove]	54
Appendix B.	Closed Issues [RFC Editor: Please remove]	54
Appendix C.	Change log [RFC Editor: Please remove]	62
Appendix D.	Example Message Formats	70
D.1.	Discovery Example	71
D.2.	Flood Example	71
D.3.	Synchronization Example	71
D.4.	Simple Negotiation Example	72
D.5.	Complete Negotiation Example	72
Appendix E.	Requirement Analysis of Discovery, Synchronization and Negotiation	73
E.1.	Requirements for Discovery	73
E.2.	Requirements for Synchronization and Negotiation Capability	75
E.3.	Specific Technical Requirements	77
Appendix F.	Capability Analysis of Current Protocols	78
Authors' Addresses	81

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:

- o 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.
- o Negotiation: a process by which two ASAs interact iteratively to agree on parameter settings that best satisfy the objectives of both ASAs.
- o 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.
- o 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.

- o Discovery Initiator: an ASA that starts discovery by sending a discovery message referring to a specific discovery objective.
- o 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.
- o Synchronization Initiator: an ASA that starts synchronization by sending a request message referring to a specific synchronization objective.
- o Synchronization Responder: a peer ASA which responds with the value of a synchronization objective.
- o Negotiation Initiator: an ASA that starts negotiation by sending a request message referring to a specific negotiation objective.
- o Negotiation Counterpart: a peer with which the Negotiation Initiator negotiates a specific negotiation objective.
- o 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.
- o 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.

- o 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.

- o 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:

- o A discovery mechanism (M_DISCOVERY, M_RESPONSE), by which an ASA can discover other ASAs supporting a given objective.
- o 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).
- o 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.
- o 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 between 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 preferable 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 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 initiator 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 = &(amp;
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

- o Name: graspy
- o Description: Python 3 implementation of GRASP core and API.
- o Maturity: Prototype code, interoperable between Windows 7 and Linux.
- o 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
- o Licensing: Simplified BSD
- o Experience: Tested on Windows, Linux and MacOS.
<https://www.cs.auckland.ac.nz/~brian/graspy/graspy.pdf>
- o 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 successfully 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 = &(amp;
  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 itself.

- 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.

- o 17. Ensure that the discovery mechanism is completely proof against loops and protected against duplicate responses.

RESOLVED: Added loop count mechanism.

- o 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.

- o 19. Should we use a text-oriented format such as JSON/CBOR instead of native binary TLV format?

RESOLVED: Yes, changed to CBOR.

- o 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.

- o 21. Rename the protocol as GRASP (GeneRiC Autonomic Signaling Protocol)?

RESOLVED: Yes, name changed.

- o 22. Does discovery mechanism scale robustly as needed? Need hop limit on relaying?

RESOLVED: Added hop limit.

- o 23. Need more details on TTL for caching discovery responses.

RESOLVED: Done.

- o 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.

- o 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.
- o 26. Add a URL type to the locator options (for security bootstrap etc.)
RESOLVED: Done, later renamed as URI.
- o 27. Security of Flood multicasts (Section 2.5.6.2).
RESOLVED: added text.
- o 28. Does ACP support secure link-local multicast?
RESOLVED by new text in the Security Considerations.
- o 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.
- o 30. Does response to discovery require randomized delays to mitigate amplification attacks?
RESOLVED: WG feedback is that it's unnecessary.
- o 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.
- o 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).
- o 33. Clarify if/when discovery needs to be repeated.
RESOLVED: Done.
- o 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.

- o 35. State that role-based authorization of ASAs is out of scope for GRASP. GRASP doesn't recognize/handle any "roles".

RESOLVED: Done.

- o 36. Reconsider CBOR definition for PEN syntax. (objective-name = text / [pen, text] ; pen = uint)

RESOLVED: See issue 29.

- o 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).

- o 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.

- o 39. Clarify that a node will contain one GRASP instance supporting multiple ASAs.

RESOLVED: Done.

- o 40. Add a "reason" code to the DECLINE option?

RESOLVED: Done.

- o 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.

- o 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.

- o 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.

- o 52. Describe in detail what is allowed and disallowed in an insecure instance of GRASP.

RESOLVED: Done.

- o 53. Tune IANA Considerations to support early assignment request.

- o 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.

- o 55. Could discovery be performed over TCP?

RESOLVED: Unicast discovery added as an option.

- o 56. Change Session-ID to 32 bits?

RESOLVED: Done.

- o 57. Add M_INVALID message?

RESOLVED: Done.

- o 58. Maximum message size?

RESOLVED by specifying default maximum message size (2048 bytes).

- o 59. Add F_NEG_DRY flag to specify a "dry run" objective?.

RESOLVED: Done.

- o 60. Change M_FLOOD syntax to associate a locator with each objective?

RESOLVED: Done.

- o 61. Is the SONN constrained instance really needed?

RESOLVED: Retained but only as an option.

- o 62. Is it helpful to tag descriptive text with message names (M_DISCOVER etc.)?

RESOLVED: Yes, done in various parts of the text.

- o 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.

- o 64. Should more security text be moved from the main text into the Security Considerations?

RESOLVED: No, on AD advice.

- o 65. Do we need to formally restrict Unicode characters allowed in objective names?

RESOLVED: No, but need to point to guidance from PRECIS WG.

- o 66. Split requirements into separate document?

RESOLVED: No, on AD advice.

- o 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 processing.

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 (corollary 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'20010db8f000baaaaf000baaaaf000baaa', 6, 49443]]
[M_RESPONSE, 13948744, h'20010db8f000baaa28ccdc4c97036781', 60000,
  [O_IPv6_LOCATOR, h'20010db8f000baaaaf000baaaaf000baaa',
  IPPROTO_TCP, 49443]]
h'85021a00d4d7485020010db8f000baaa28ccdc4c9703678119ea6084186750
  20010db8f000baaaaf000baaaaf000baaa0619c123'
```

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'83031a000c3ffd8463455833030682634e5a44182f'
```

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'83031a00d214628463455833030682634e5a4419019a'
```

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'83051a00d214628463455833030682634e5a441850'
```

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'83051a00d214628463455833030582634e5a44190133'
```

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'83061a00d2146282186672496e7375666696369656e742066756e6473'
```

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.

- o 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
Internet-Draft
Intended status: Informational
Expires: May 27, 2019

M. Behringer, Ed.
B. Carpenter
Univ. of Auckland
T. Eckert
Futurewei Technologies Inc.
L. Ciavaglia
Nokia
J. Nobre
University of Vale do Rio dos Sinos
November 23, 2018

A Reference Model for Autonomic Networking
draft-ietf-anima-reference-model-10

Abstract

This document describes a reference model for Autonomic Networking for managed networks. It defines the behaviour of an autonomic node, how the various elements in an autonomic context work together, and how autonomic services can use the infrastructure.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 27, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. The Network View	4
3. The Autonomic Network Element	5
3.1. Architecture	5
3.2. The Adjacency Table	6
3.3. State Machine	8
3.3.1. State 1: Factory Default	8
3.3.2. State 2: Enrolled	9
3.3.3. State 3: In ACP	9
4. The Autonomic Networking Infrastructure	10
4.1. Naming	10
4.2. Addressing	10
4.3. Discovery	12
4.4. Signaling Between Autonomic Nodes	12
4.5. Routing	13
4.6. The Autonomic Control Plane	13
4.7. Information Distribution (*)	13
5. Security and Trust Infrastructure	14
5.1. Public Key Infrastructure	14
5.2. Domain Certificate	14
5.3. The MASA	15
5.4. Sub-Domains (*)	15
5.5. Cross-Domain Functionality (*)	15
6. Autonomic Service Agents (ASA)	15
6.1. General Description of an ASA	15
6.2. ASA Life-Cycle Management	17
6.3. Specific ASAs for the Autonomic Network Infrastructure	18
6.3.1. The enrollment ASAs	18
6.3.2. The ACP ASA	19
6.3.3. The Information Distribution ASA (*)	19
7. Management and Programmability	19
7.1. Managing a (Partially) Autonomic Network	19
7.2. Intent (*)	20
7.3. Aggregated Reporting (*)	21
7.4. Feedback Loops to NOC (*)	21
7.5. Control Loops (*)	22
7.6. APIs (*)	22
7.7. Data Model (*)	23
8. Coordination Between Autonomic Functions (*)	24

8.1. The Coordination Problem (*)	24
8.2. A Coordination Functional Block (*)	25
9. Security Considerations	25
9.1. Protection Against Outsider Attacks	26
9.2. Risk of Insider Attacks	27
10. IANA Considerations	27
11. Acknowledgements	28
12. Contributors	28
13. References	28
13.1. Normative References	28
13.2. Informative References	28
Authors' Addresses	30

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, as well as a high level reference model. [RFC7576] provides a gap analysis between traditional and autonomic approaches.

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.

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.

For example, it is possible in an existing, non-autonomic network to enrol devices in a traditional way, to bring up a trust infrastructure with certificates. This trust infrastructure could then be used to automatically bring up an Autonomic Control Plane (ACP), and run traditional network operations over the secure and self-healing ACP. See [I-D.ietf-anima-stable-connectivity] for a description of this use case.

The scope of this model is therefore limited to networks that are to some extent managed by skilled human operators, loosely referred to as "professionally managed" networks. Unmanaged networks raise additional security and trust issues that this model does not cover.

This document describes a first, simple, implementable phase of an Autonomic Networking solution. It is expected that the experience from this phase will be used in defining updated and extended specifications over time. Some topics are considered architecturally

in this document, but are not yet reflected in the implementation specifications. They are marked with an (*).

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.

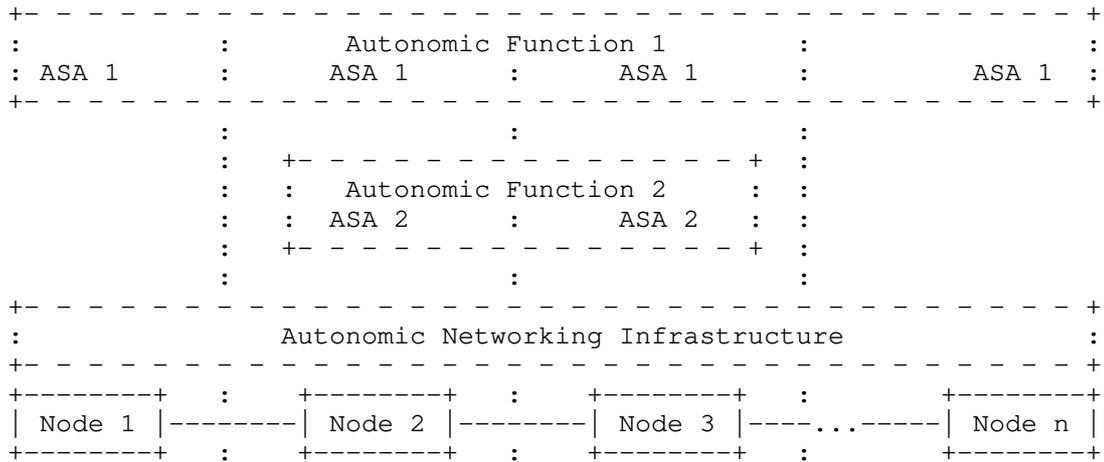


Figure 1: High level view of an Autonomic Network

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. ASAs may be standalone, or use other ASAs in a hierarchical way.

The Autonomic Networking Infrastructure (ANI) therefore is the foundation for autonomic functions.

3. The Autonomic Network Element

This section explains the general architecture of an Autonomic Network Element (Section 3.1), how it tracks its surrounding environment in an Adjacency Table (Section 3.2), and the state machine which defines the behaviour of the network element (Section 3.3), based on that adjacency table.

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 (see Section 7.2), and feedback loops. 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.

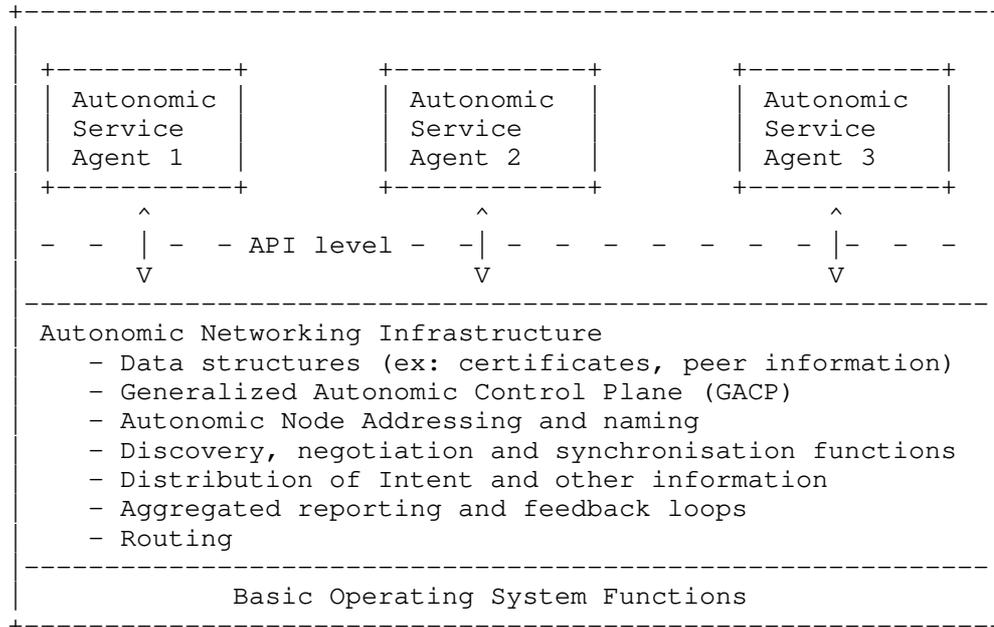


Figure 2: Model of an autonomic node

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). It contains addressing and naming of autonomic nodes, discovery, negotiation and synchronisation functions, distribution of information, reporting and feedback loops, as well as routing inside the Autonomic Control Plane.

The Generalized Autonomic Control Plane (GACP) is the summary of all interactions of the Autonomic Networking Infrastructure with other nodes and services. A specific implementation of the GACP is referred to here as the Autonomic Control Plane (ACP), and described in [I-D.ietf-anima-autonomic-control-plane].

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, which should 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.

3.2. The Adjacency Table

Autonomic Networking is based on direct interactions between devices of a domain. The Autonomic Control Plane (ACP) is normally constructed 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 level of the adjacent autonomic nodes.

The adjacency table is fed by the following inputs:

- o 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. The related standards track documents ([I-D.ietf-anima-grasp], [I-D.ietf-anima-bootstrapping-keyinfra], [I-D.ietf-anima-autonomic-control-plane]) describe in detail how link local discovery is used.
- o Vendor re-direct: A new device may receive information on where its home network is through a vendor based Manufacturer Authorized Signing Authority (MASA, see Section 5.3) re-direct; this is typically a routable address.
- o 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 mechanisms 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 appendix A and B of [I-D.ietf-anima-bootstrapping-keyinfra].

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 re-direct via a vendor MASA, which will be entered into the adjacency table (see second bullet above). See [I-D.ietf-anima-bootstrapping-keyinfra] for details.
- o If the adjacent node has the same domain, it will authenticate that adjacent node and, if successful, establish the Autonomic Control Plane (ACP). See [I-D.ietf-anima-autonomic-control-plane].
- o Once the node is part of the ACP of a domain, it will use GRASP [I-D.ietf-anima-grasp] to find Registrar(s) of its domain and potentially other services.
- o If the node is part of an ACP and has discovered at least one Registrar in its domain via GRASP, it will start the "join

assistant" ASA, and act as a join assistant for neighboring nodes that need to be bootstrapped. See Section 6.3.1.2 for details.

- 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 autonomic domain 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.

3.3. State Machine

Autonomic Networking applies during the full life-cycle of a node. This section describes a state machine of an autonomic node, throughout its life.

A device is normally expected to store its domain specific identity, the LDevID (see Section 5.2), in persistent storage, to be available after a powercycle event. For device types that cannot store the LDevID in persistent storage, a powercycle event is effectively equivalent to a factory reset.

3.3.1. State 1: Factory Default

An autonomic node leaves the factory in this state. In this state, the node has no domain specific configuration, specifically no LDevID, and could be used in any particular target network. It does however have a vendor/manufacturer specific ID, the IDevID [IDevID]. Nodes without IDevID cannot be autonomically and securely enrolled into a domain; they require manual pre-staging, in which case the pre-staging takes them directly to state 2.

Transitions:

- o Bootstrap event: The device enrolls into a domain; as part of this process it receives a domain identity (LDevID). If enrollment is successful, the next state is state 2. See [I-D.ietf-anima-bootstrapping-keyinfra] Section 3 for details on enrollment.
- o Powercycle event: The device loses all state tables. It remains in state: 1.

3.3.2. State 2: Enrolled

An autonomic node is in the state "enrolled" if it has a domain identity (LDevID), and has currently no ACP channel up. It may have further configuration or state, for example if it had been in state 3 before, but lost all its ACP channels. The LDevID can only be removed from a device through a factory reset, which also removes all other state from the device. This ensures that a device has no stale domain specific state when entering the "enrolled" state from state 1.

Transitions:

- o Joining ACP: The device establishes an ACP channel to an adjacent device. See [I-D.ietf-anima-autonomic-control-plane] for details. Next state: 3.
- o Factory reset: A factory reset removes all configuration and the domain identity (LDevID) from the device. Next state: 1.
- o Powercycle event: The device loses all state tables, but not its domain identity (LDevID). it remains in state: 2.

3.3.3. State 3: In ACP

In this state, the autonomic node has at least one ACP channel to another device. The node can now participate in further autonomic transactions, such as starting autonomic service agents (e.g., it must now enable the join assistant ASA, to help other devices to join the domain. Other conditions may apply to such interactions, for example to serve as a join assistant, the device must first discover a bootstrap Registrar.

Transitions:

- o Leaving ACP: The device drops the last (or only) ACP channel to an adjacent device. Next state: 2.

- o Factory reset: A factory reset removes all configuration and the domain identity (LDevID) from the device. Next state: 1.
- o Powercycle event: The device loses all state tables, but not its domain identity (LDevID). Next state: 2.

4. The Autonomic Networking Infrastructure

The Autonomic Networking Infrastructure provides a layer of common functionality across an Autonomic Network. It provides the elementary functions and services, as well as extensions. An Autonomic Function, comprising of Autonomic Service Agents on nodes, uses the functions described in this section.

4.1. Naming

Inside a domain, each autonomic device should be assigned a unique name. The naming scheme should be consistent within a domain. Names are typically assigned by a Registrar at bootstrap time and persistent over the lifetime of the device. All Registrars in a domain must follow the same naming scheme.

In the absence of a domain specific naming scheme, a default naming scheme should use the same logic as the addressing scheme discussed in [I-D.ietf-anima-autonomic-control-plane]. The device name is then composed of a Registrar ID (for example taking a MAC address of the Registrar) and a device number. An example name would then look like this:

0123-4567-89ab-0001

The first three fields are the MAC address, the fourth field is the sequential number for the device.

4.2. Addressing

Autonomic Service Agents (ASAs) need to communicate with each other, using the autonomic addressing of the Autonomic Networking Infrastructure of the node they reside on. This section describes the addressing approach of the Autonomic Networking Infrastructure, used by ASAs.

Addressing approaches for the data plane of the network are outside the scope of this document. These addressing approaches 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.ietf-anima-prefix-management].

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 Stability: The addressing scheme should be as stable as possible. However, implementations need to be able to recover from unexpected address changes.
- o Support for virtualization: Autonomic functions can exist either at the level of the physical network and physical devices, or at the level of virtual machines, containers and networks. In particular, Autonomic Nodes may support Autonomic Service Agents in virtual entities. The infrastructure, including the addressing scheme, should be able to 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 proposed addressing scheme is described in the document "An Autonomic Control Plane" ([I-D.ietf-anima-autonomic-control-plane]).

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 information distribution, Section 4.7).

Phase 1 of Autonomic Networking uses GRASP for discovery, described in [I-D.ietf-anima-grasp].

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 Autonomic Signaling Protocol (GRASP)" [I-D.ietf-anima-grasp] describes more detailed requirements for discovery, negotiation and synchronization in an autonomic network. It also defines a protocol, GRASP, for this purpose, including an integrated but optional discovery protocol.

GRASP is normally expected to run inside the Autonomic Control Plane (ACP; see Section 4.6) and to depend on the ACP for security. It may run insecurely for a short time during bootstrapping.

An autonomic node will normally run a single instance of GRASP, used by multiple ASAs. However, scenarios where multiple instances of GRASP run in a single node, perhaps with different security properties, are not excluded.

4.5. Routing

All autonomic nodes in a domain must be able to communicate with each other, and later phases also 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.

The routing protocol is defined in the ACP document [I-D.ietf-anima-autonomic-control-plane].

4.6. The Autonomic Control Plane

The "Autonomic Control Plane" carries the control protocols in an autonomic network. In the architecture described here, it is implemented as an overlay network. The document "An Autonomic Control Plane" ([I-D.ietf-anima-autonomic-control-plane]) describes the implementation details suggested here. This document uses the term "overlay" to mean a set of point-to-point adjacencies congruent with the underlying interconnection topology. The terminology may not be aligned with a common usage of the "overlay" term in routing context. See [I-D.ietf-anima-stable-connectivity] for uses cases for the ACP.

4.7. Information Distribution (*)

Certain forms of information require distribution across an autonomic domain. The distribution of information runs inside the Autonomic Control Plane. For example, Intent is distributed across an autonomic domain, as explained in [RFC7575].

Intent is the policy language of an Autonomic Network, see also Section 7.2. It is a high level policy, and should change only infrequently (order of days). Therefore, information such as Intent should be simply flooded to all nodes in an autonomic domain, and there is currently no perceived need to have more targeted distribution methods. Intent is also expected to be monolithic, and flooded as a whole. One possible method for distributing Intent, as well as other forms of data, is discussed in [I-D.liu-anima-grasp-distribution]. Intent and information distribution are not part of phase 1 of ANIMA.

5. 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, with the exception of setting up a PKI infrastructure.

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. In this first phase of autonomic networking, a device is either within the trust domain and fully trusted, or outside the trust domain and fully untrusted.

The default method to automatically bring up a trust infrastructure is defined in the document "Bootstrapping Key Infrastructures" [I-D.ietf-anima-bootstrapping-keyinfra]. The ASAs required for this enrollment process are described in Section 6.3. An autonomic node must implement the enrollment and join assistant ASAs. The registrar ASA may be implemented only on a sub-set of nodes.

5.1. 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.

5.2. Domain Certificate

Each device in an autonomic domain uses a domain certificate (LDevID) to prove its identity. A new device uses its manufacturer provided certificate (IDevID) during bootstrap, to obtain a domain

certificate. [I-D.ietf-anima-bootstrapping-keyinfra] describes how a new device receives a domain certificate, and the certificate format.

5.3. The MASA

The Manufacturer Authorized Signing Authority (MASA) is a trusted service for bootstrapping devices. The purpose of the MASA is to provide ownership tracking of devices in a domain. The MASA provides audit, authorization, and ownership tokens to the registrar during the bootstrap process to assist in the authentication of devices attempting to join an Autonomic Domain, and to allow a joining device to validate whether it is joining the correct domain. The details for MASA service, security, and usage are defined in [I-D.ietf-anima-bootstrapping-keyinfra].

5.4. Sub-Domains (*)

By default, sub-domains are treated as different domains. This implies no trust between a domain and its sub-domains, and no trust between sub-domains of the same domain. Specifically, no ACP is built, and Intent is valid only for the domain it is defined for explicitly.

In phase 2 of ANIMA, alternative trust models should be defined, for example to allow full or limited trust between domain and sub-domain.

5.5. Cross-Domain Functionality (*)

By default, different domains do not interoperate, no ACP is built and no trust is implied between them.

In the future, models can be established where other domains can be trusted in full or for limited operations between the domains.

6. Autonomic Service Agents (ASA)

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

6.1. General Description of an ASA

An Autonomic Service Agent (ASA) is defined in [RFC7575] as "An agent implemented on an autonomic node that implements an autonomic function, either in part (in the case of a distributed function) or whole." Thus it is a process that makes use of the features provided by the ANI to achieve its own goals, usually including interaction with other ASAs via the GRASP protocol [I-D.ietf-anima-grasp] or otherwise. Of course it also interacts with the specific targets of

its function, using any suitable mechanism. Unless its function is very simple, the ASA will need to handle overlapping asynchronous operations. It may therefore be a quite complex piece of software in its own right, forming part of the application layer above the ANI. ASA design guidelines are available in [I-D.carpenter-anima-asa-guidelines].

Thus we can distinguish at least three classes of ASAs:

- o Simple ASAs with a small footprint that could run anywhere.
- o Complex, possibly multi-threaded ASAs that have a significant resource requirement and will only run on selected nodes.
- o A few 'infrastructure ASAs' that use basic ANI features in support of the ANI itself, which must run in all autonomic nodes. These are outlined in the following sections.

Autonomic nodes, and therefore their ASAs, know their own capabilities and restrictions, derived from hardware, firmware or pre-installed software: They are "self-aware".

The role of an autonomic node depends on Intent and on the surrounding network behaviors, which may include forwarding behaviors, aggregation properties, topology location, bandwidth, tunnel or translation properties, etc. For example, a node may decide to act as a backup node for a neighbor, if its capabilities allow it to do so.

Following an initial discovery phase, the node properties and those of its neighbors are the foundation of the behavior of a specific node. A node and its ASAs have no pre-configuration for the particular network in which they are installed.

Since all ASAs will interact with the ANI, they will depend on appropriate application programming interfaces (APIs). It is desirable that ASAs are portable between operating systems, so these APIs need to be universal. An API for GRASP is described in [I-D.ietf-anima-grasp-api].

ASAs will in general be designed and coded by experts in a particular technology and use case, not by experts in the ANI and its components. Also, they may be coded in a variety of programming languages, in particular including languages that support object constructs as well as traditional variables and structures. The APIs should be designed with these factors in mind.

It must be possible to run ASAs as non-privileged (user space) processes except for those (such as the infrastructure ASAs) that necessarily require kernel privilege. Also, it is highly desirable that ASAs can be dynamically loaded on a running node.

Since autonomic systems must be self-repairing, it is of great importance that ASAs are coded using robust programming techniques. All run-time error conditions must be caught, leading to suitable minimally disruptive recovery actions, also considering a complete restart of the ASA. Conditions such as discovery failures or negotiation failures must be treated as routine, with the ASA retrying the failed operation, preferably with an exponential back-off in the case of persistent errors. When multiple threads are started within an ASA, these threads must be monitored for failures and hangups, and appropriate action taken. Attention must be given to garbage collection, so that ASAs never run out of resources. There is assumed to be no human operator - again, in the worst case, every ASA must be capable of restarting itself.

ASAs will automatically benefit from the security provided by the ANI, and specifically by the ACP and by GRASP. However, beyond that, they are responsible for their own security, especially when communicating with the specific targets of their function. Therefore, the design of an ASA must include a security analysis beyond 'use ANI security.'

6.2. ASA Life-Cycle Management

ASAs operating on a given ANI may come from different providers and pursue different objectives. Management of ASAs and its interactions with the ANI should follow the same operating principles, hence comply to a generic life-cycle management model.

The ASA life-cycle provides standard processes to:

- o install ASA: copy the ASA code onto the node and start it,
- o deploy ASA: associate the ASA instance with a (some) managed network device(s) (or network function),
- o control ASA execution: when and how an ASA executes its control loop.

The life-cycle will cover the sequential states below: Installation, Deployment, Operation and the transitional states in-between. This Life-Cycle will also define which interactions ASAs have with the ANI in between the different states. The noticeable interactions are:

- o Self-description of ASA instances at the end of deployment: its format needs to define the information required for the management of ASAs by ANI entities
- o Control of ASA control-loop during the operation: a signaling has to carry formatted messages to control ASA execution (at least starting and stopping the control loop)

6.3. Specific ASAs for the Autonomic Network Infrastructure

The following functions provide essential, required functionality in an autonomic network, and are therefore mandatory to implement on unconstrained autonomic nodes. They are described here as ASAs that include the underlying infrastructure components, but implementation details might vary.

The first three together support the trust enrollment process described in Section 5. For details see [I-D.ietf-anima-bootstrapping-keyinfra].

6.3.1. The enrollment ASAs

6.3.1.1. The Pledge ASA

This ASA includes the function of an autonomic node that bootstraps into the domain with the help of an join assistant ASA (see below). Such a node is known as a Pledge during the enrollment process. This ASA must be installed by default on all nodes that require an autonomic zero-touch bootstrap.

6.3.1.2. The Join Assistant ASA

This ASA includes the function of an autonomic node that helps a non-enrolled, adjacent devices to enroll into the domain. This ASA must be installed on all nodes, although only one join assistant needs to be active on a given LAN. See also [I-D.ietf-anima-bootstrapping-keyinfra].

6.3.1.3. The Join Registrar ASA

This ASA includes the join registrar function in an autonomic network. This ASA does not need to be installed on all nodes, but only on nodes that implement the Join Registrar function.

6.3.2. The ACP ASA

This ASA includes the ACP function in an autonomic network. In particular it acts to discover other potential ACP nodes, and to support the establishment and teardown of ACP channels. This ASA must be installed on all nodes. For details see Section 4.6 and [I-D.ietf-anima-autonomic-control-plane].

6.3.3. The Information Distribution ASA (*)

This ASA is currently out of scope in ANIMA, and provided here only as background information.

This ASA includes the information distribution function in an autonomic network. In particular it acts to announce the availability of Intent and other information to all other autonomic nodes. This ASA does not need to be installed on all nodes, but only on nodes that implement the information distribution function. For details see Section 4.7.

Note that information distribution can be implemented as a function in any ASA. See [I-D.liu-anima-grasp-distribution] for more details on how information is suggested to be distributed.

7. Management and Programmability

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

7.1. Managing a (Partially) Autonomic Network

Autonomic management usually co-exists with traditional management methods in most networks. Thus, autonomic behavior will be defined for individual functions in most environments. Examples for overlap are:

- o Autonomic functions can use traditional methods and protocols (e.g., SNMP and NETCONF) to perform management tasks, inside and outside the ACP;
- o Autonomic functions can conflict with behavior enforced by the same traditional methods and protocols;
- o Traditional functions can use the ACP, for example if reachability on the data plane is not (yet) established.

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.

7.2. Intent (*)

Intent is not covered in the current implementation specifications. This section discusses a topic for further research.

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

Intent can be described 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.

Intent can be refined to lower level policies using different approaches. This is expected in order to adapt the Intent to the capabilities of managed devices. Intent may contain role or function information, which can be translated to specific nodes [RFC7575]. One of the possible refinements of the Intent is using Event-Condition-Action (ECA) rules.

Different parameters may be configured for Intent. 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.

Intent is discussed in more detail in [I-D.du-anima-an-intent]. Intent as well as other types of information are distributed via GRASP, see [I-D.liu-anima-grasp-distribution].

7.3. Aggregated Reporting (*)

Aggregated reporting is not covered in the current implementation specifications. This section discusses a topic for further research.

An 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].

Multiple simultaneous events can occur in an autonomic network in the same way they can happen in a traditional network. However, when reporting to a human administrator, such events should be aggregated to avoid notifications 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 to maintain a network-wide target which is described in the autonomic Intent.

Reporting in an autonomic network may be at the same abstraction level as Intent. In this context, the aggregated view of 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.

7.4. 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 must be prompted with a default action, and has the possibility to acknowledge or override the proposed default action.

Uni-directional notifications to the NOC, that do not propose any default action, and do not allow an override as part of the transaction are considered like traditional notification services, such as syslog. They are expected to co-exist with autonomic methods, but are not covered in this draft.

7.5. Control Loops (*)

Control loops are not covered in the current implementation specifications. This section discusses a topic for further research.

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 network 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.

7.6. APIs (*)

APIs are not covered in the current implementation specifications. This section discusses a topic for further research.

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 the semantics of data models. 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

7.7. Data Model (*)

Data models are not covered in the current implementation specifications. This section discusses a topic for further research.

The following definitions are adapted from [I-D.ietf-supra-generic-policy-data-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 Model-Reference Adaptive Control Loop (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.

8. Coordination Between Autonomic Functions (*)

Coordination between autonomic functions is not covered in the current implementation specifications. This section discusses a topic for further research.

8.1. 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:

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

- o 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.
- o 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.

- o 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 exist 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.

8.2. 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:

- o 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.

9. Security Considerations

In this section we distinguish outsider and insider attacks. In an outsider attack all network elements and protocols are securely managed and operating, and an outside attacker can sniff packets in transit, inject and replay packets. In an insider attack, the attacker has access to an autonomic node or other means (e.g. remote code execution in the node by exploiting ACP-independent vulnerabilities in the node platform) to produce arbitrary payloads on the protected ACP channels.

If a system has vulnerabilities in the implementation or operation (configuration), an outside attacker can exploit such vulnerabilities to become an insider attacker.

9.1. Protection Against Outsider Attacks

Here, we assume that all systems involved in an autonomic network are secured and operated according to best current practices. These protection methods comprise traditional security implementation and operation methods (such as code security, strong randomization algorithms, strong passwords, etc.) as well as mechanisms specific to an autonomic network (such as a secured MASA service).

Traditional security methods for both implementation and operation are outside scope for this document.

AN specific protocols and methods must also follow traditional security methods, in that all packets that can be sniffed or injected by an outside attacker are:

- o protected against modification.
- o authenticated.
- o protected against replay attacks.
- o confidentiality protected (encrypted).
- o and that the AN protocols are robust against packet drops and man-in-the-middle attacks.

How these requirements are met is covered in the AN standards track documents that define the methods used, specifically [I-D.ietf-anima-bootstrapping-keyinfra], [I-D.ietf-anima-grasp], and [I-D.ietf-anima-autonomic-control-plane].

Most AN messages run inside the cryptographically protected ACP. The unprotected AN messages outside the ACP are limited to a simple discovery method, defined in Section 2.5.2 of [I-D.ietf-anima-grasp]: The "Discovery Unsolicited Link-Local (DULL)" message, with detailed rules on its usage.

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.

9.2. Risk of Insider Attacks

An autonomic network consists of autonomic devices that form a distributed self-managing system. Devices within a domain have credentials issued from a common trust anchor and can use them to create mutual trust. This means that any device inside a trust domain can by default use all distributed functions in the entire autonomic domain in a malicious way.

If an autonomic node or protocol has vulnerabilities or is not securely operated, an outside attacker has the following generic ways to take control of an autonomic network:

- o Introducing a fake device into the trust domain, by subverting the authentication methods. This depends on the correct specification, implementation and operation of the AN protocols.
- o Subverting a device which is already part of a trust domain, and modifying its behavior. This threat is not specific to the solution discussed in this document, and applies to all network solutions.
- o Exploiting potentially yet unknown protocol vulnerabilities in the AN or other protocols. Also this is a generic threat that applies to all network solutions.

The above threats are in principle comparable to other solutions: In the presence of design, implementation or operational errors, security is no longer guaranteed. However, the distributed nature of AN, specifically the Autonomic Control Plane, increases the threat surface significantly. For example, a compromised device may have full IP reachability to all other devices inside the ACP, and can use all AN methods and protocols.

For the next phase of the ANIMA work it is therefore recommended to introduce a sub-domain security model, to reduce the attack surface and not expose a full domain to a potential intruder. Furthermore, additional security mechanisms on the ASA level should be considered for high-risk autonomic functions.

10. IANA Considerations

This document requests no action by IANA.

11. Acknowledgements

Many people have provided feedback and input to this document: Sheng Jiang, Roberta Maglione, Jonathan Hansford, Jason Coleman, Artur Hecker. Useful reviews were made by Joel Halpern, Radia Perlman, Tianran Zhou and Christian Hopps.

12. Contributors

Significant contributions to this document have been made by John Strassner and Bing Liu from Huawei, and Pierre Peloso from Nokia.

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-18 (work in progress), August 2018.
- [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-17 (work in progress), November 2018.
- [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 Standard, , "IEEE 802.1AR Secure Device Identifier", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.

13.2. Informative References

- [I-D.carpenter-anima-asa-guidelines]
Carpenter, B., Ciavaglia, L., Jiang, S., and P. Pierre, "Guidelines for Autonomic Service Agents", draft-carpenter-anima-asa-guidelines-05 (work in progress), June 2018.
- [I-D.du-anima-an-intent]
Du, Z., Jiang, S., Nobre, J., Ciavaglia, L., and M. Behringer, "ANIMA Intent Policy and Format", draft-du-anima-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-02 (work in progress), June 2018.
- [I-D.ietf-anima-prefix-management]
Jiang, S., Du, Z., and B. Carpenter, "Autonomic IPv6 Edge Prefix Management in Large-scale Networks", draft-ietf-anima-prefix-management-07 (work in progress), December 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-10 (work in progress), February 2018.
- [I-D.ietf-supra-generic-policy-data-model]
Halpern, J. and J. Strassner, "Generic Policy Data Model for Simplified Use of Policy Abstractions (SUPA)", draft-ietf-supra-generic-policy-data-model-04 (work in progress), June 2017.
- [I-D.liu-anima-grasp-distribution]
Liu, B., Jiang, S., Xiao, X., Hecker, A., and Z. Despotovic, "Information Distribution in Autonomic Networking", draft-liu-anima-grasp-distribution-09 (work in progress), October 2018.
- [Meyer97] Meyer, B., "Object-Oriented Software Construction (2nd edition)", Prentice-Hall, ISBN 978-0136291558, 1997.
- [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>>.

Authors' Addresses

Michael H. Behringer (editor)

Email: Michael.H.Behringer@gmail.com

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

Email: brian.e.carpenter@gmail.com

Toerless Eckert
Futurewei Technologies Inc.
2330 Central Expy
Santa Clara 95050
USA

Email: tte@cs.fau.de

Laurent Ciavaglia
Nokia
Villarceaux
Nozay 91460
FR

Email: laurent.ciavaglia@nokia.com

Jeferson Campos Nobre
University of Vale do Rio dos Sinos
Av. Unisinos, 950
Sao Leopoldo 91501-970
Brazil

Email: jcnobre@unisinos.br

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 28, 2018

B. Carpenter
Univ. of Auckland
B. Liu, Ed.
Huawei Technologies
W. Wang
X. Gong
BUPT University
November 24, 2017

Generic Autonomic Signaling Protocol Application Program Interface
(GRASP API)
draft-liu-anima-grasp-api-06

Abstract

This document is a conceptual outline of the application programming interface (API) of the Generic Autonomic Signaling Protocol (GRASP). Such an API is needed for Autonomic Service Agents (ASA) calling the GRASP protocol module to exchange autonomic network messages with other ASAs.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 28, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. GRASP API for ASA	3
2.1. Design Principles	3
2.2. Asynchronous Operations	4
2.3. API definition	6
2.3.1. Parameters and data structures	6
2.3.2. Registration	9
2.3.3. Discovery	11
2.3.4. Negotiation	12
2.3.5. Synchronization and Flooding	17
2.3.6. Invalid Message Function	20
3. Example Logic Flows	21
4. Security Considerations	21
5. IANA Considerations	21
6. Acknowledgements	21
7. References	21
7.1. Normative References	21
7.2. Informative References	22
Appendix A. Error Codes	22
Appendix B. Change log [RFC Editor: Please remove]	23
Authors' Addresses	24

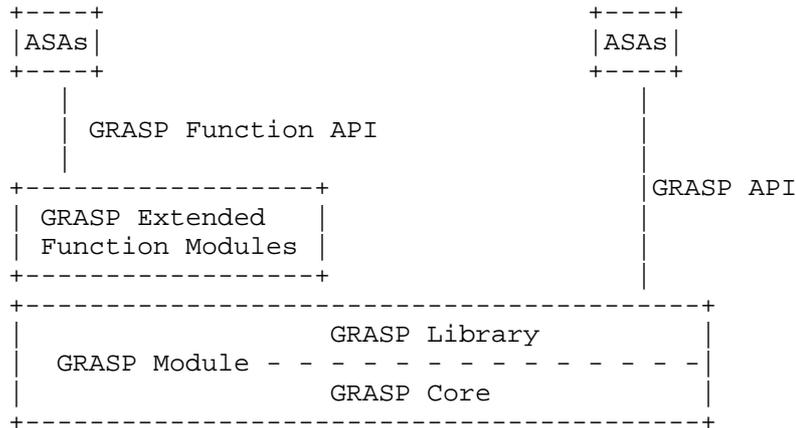
1. Introduction

As defined in [I-D.ietf-anima-reference-model], the Autonomic Service Agent (ASA) is the atomic entity of an autonomic function; and it is instantiated on autonomic nodes. When ASAs communicate with each other, they should use the Generic Autonomic Signaling Protocol (GRASP) [I-D.ietf-anima-grasp].

As the following figure shows, GRASP could contain two major sub-layers. The bottom is the GRASP base protocol module, which is only responsible for sending and receiving GRASP messages and maintaining shared data structures. The upper layer is some extended functions based upon GRASP basic protocol. For example, [I-D.liu-anima-grasp-distribution] describes a possible extended function.

It is desirable that ASAs can be designed as portable user-space programs using a portable API. In many operating systems, the GRASP

module will therefore be split into two layers, one being a library that provides the API and the other being core code containing common components such as multicast handling and the discovery cache. The details of this are system-dependent. In particular, the GRASP library might need to communicate with the GRASP core via an inter-process communication (IPC) mechanism.



Both the GRASP library and the extended function modules should be available to the ASAs. Thus, there needs to be two sub-sets of API. However, since the extended functions are expected to be added in an incremental manner, it is inappropriate to define the function APIs in a single document. This document only defines the base GRASP API.

Note that a very simple autonomic node might contain only a single ASA in addition to the autonomic infrastructure components described in [I-D.ietf-anima-bootstrapping-keyinfra] and [I-D.ietf-anima-autonomic-control-plane]. Such a node might directly integrate GRASP in its autonomic code and therefore not require this API to be installed.

This document gives a conceptual outline of the API. It is not a formal specification for any particular programming language or operating system, and it is expected that details will be clarified in individual implementations.

2. GRASP API for ASA

2.1. Design Principles

The assumption of this document is that any Autonomic Service Agent (ASA) needs to call a GRASP module that handles protocol details (security, sending and listening for GRASP messages, waiting, caching

discovery results, negotiation looping, sending and receiving synchronization data, etc.) but understands nothing about individual objectives. So this is a high level abstract API for use by ASAs. Individual language bindings should be defined in separate documents.

An assumption of this API is that ASAs may fall into various classes:

- o ASAs that only use GRASP for discovery purposes.
- o ASAs that use GRASP negotiation but only as an initiator (client).
- o ASAs that use GRASP negotiation but only as a responder.
- o ASAs that use GRASP negotiation as an initiator or responder.
- o ASAs that use GRASP synchronization but only as an initiator (recipient).
- o ASAs that use GRASP synchronization but only as a responder and/or flooder.
- o ASAs that use GRASP synchronization as an initiator, responder and/or flooder.

The API also assumes that one ASA may support multiple objectives. Nothing prevents an ASA from supporting some objectives for synchronization and others for negotiation.

The API design assumes that the operating system and programming language provide a mechanism for simultaneous asynchronous operations. This is discussed in detail in Section 2.2.

This is a preliminary version. A few gaps exist:

- o Authorization of ASAs is out of scope.
- o User-supplied explicit locators for an objective are not supported.
- o The Rapid mode of GRASP is not supported.

2.2. Asynchronous Operations

GRASP includes asynchronous operations and wait states. Most ASAs will need to support several simultaneous operations; for example an ASA might need to negotiate one objective with a peer while discovering and synchronizing a different objective with a different peer. Alternatively, an ASA which acts as a resource manager might

need to run simultaneous negotiations for a given objective with multiple different peers. Thus, both the GRASP core and most ASAs need to support asynchronous operations. Depending on both the operating system and the programming language in use, there are two main techniques for such parallel operations: multi-threading, or a polling or 'event loop' structure.

In multi-threading, the operating system and language will provide the necessary support for asynchronous operations, including creation of new threads, context switching between threads, queues, locks, and implicit wait states. In this case, all API calls can be treated naturally as synchronous, even if they include wait states, blocking and queueing.

In an event loop implementation, synchronous blocking calls are not acceptable. Therefore all calls must be non-blocking, and the main loop will support multiple GRASP sessions in parallel by repeatedly checking each one for a change of state. To facilitate this, the API implementation will provide non-blocking versions of all the functions that otherwise involve blocking and queueing. In these calls, a 'noReply' code will be returned by each call instead of blocking, until such time as the event for which it is waiting has occurred. Thus, for example, `discover()` would return "noReply" instead of waiting until discovery has succeeded or timed out. The `discover()` call would be repeated in every cycle of the main loop until it completes. A 'session_nonce' parameter (described below) is used to distinguish simultaneous GRASP sessions from each other, so that any number of sessions may proceed in parallel.

The following calls involve waiting for a remote operation, so they use this mechanism:

```
discover()

request_negotiate()

negotiate_step()

listen_negotiate()

synchronize()
```

In all these calls, the 'session_nonce' is a read/write parameter. On the first call, it is set to a null value, and the API returns the 'noReply' code and a non-null value. This value must be used in all subsequent calls. By this mechanism, multiple overlapping sessions can be distinguished.

2.3. API definition

2.3.1. Parameters and data structures

This section describes parameters and data structures used in multiple API calls.

2.3.1.1. Errorcode

All functions in the API have an unsigned 'errorcode' integer as their return value (the first returned value in languages that allow multiple returned parameters). An errorcode of zero indicates success. Any other value indicates failure of some kind. The first three errorcodes have special importance:

1. Declined: used to indicate that the other end has sent a GRASP Negotiation End message (M_END) with a Decline option (O_DECLINE).
2. No reply: used in non-blocking calls to indicate that the other end has sent no reply so far (see Section 2.2).
3. Unspecified error: used when no more specific error code applies.

Appendix A gives a full list of currently defined error codes.

2.3.1.2. Timeout

Wherever a 'timeout' parameter appears, it is an integer expressed in milliseconds. If it is zero, the GRASP default timeout (GRASP_DEF_TIMEOUT, see [I-D.ietf-anima-grasp]) will apply. If no response is received before the timeout expires, the call will fail unless otherwise noted.

2.3.1.3. Objective

An 'objective' parameter is a data structure with the following components:

- o name (UTF-8 string) - the objective's name
- o neg (Boolean flag) - True if objective supports negotiation (default False)
- o synch (Boolean flag) - True if objective supports synchronization (default False)

- o `dry` (Boolean flag) - True if objective supports dry-run synchronization (default False)
 - * Note 1: All objectives are assumed to support discovery, so there is no Boolean for that.
 - * Note 2: Only one of 'synch' or 'neg' may be True.
 - * Note 3: 'dry' must not be True unless 'neg' is also True.
- o `loop_count` (integer) - Limit on negotiation steps etc. (default GRASP_DEF_LOOPCT, see [I-D.ietf-anima-grasp])
- o `value` - a specific data structure expressing the value of the objective. The format is language dependent, with the constraint that it can be validly represented in CBOR (default integer = 0).

An essential requirement for all language mappings and all implementations is that, regardless of what other options exist for a language-specific representation of the value, there is always an option to use a CBOR byte string as the value. The API will then wrap this byte string in CBOR Tag 24 for transmission via GRASP, and unwrap it after reception.

An example data structure definition for an objective in the C language is:

```
typedef struct {
    char *name;
    uint8_t flags;           // flag bits as defined by GRASP
    int loop_count;
    int value_size;        // size of value
    uint8_t cbor_value[];  // CBOR bytestring of value
} objective;
```

An example data structure definition for an objective in the Python language is:

```
class objective:
    """A GRASP objective"""
    def __init__(self, name):
        self.name = name      #Unique name, string
        self.neg = False     #True if objective supports negotiation
        self.dry = False     #True if objective supports dry-run negotiation
        self.synch = False   #True if objective supports synch
        self.loop_count = GRASP_DEF_LOOPCT #Default starting value
        self.value = 0       #Place holder; any valid Python object
```

2.3.1.4. ASA_locator

An 'ASA_locator' parameter is a data structure with the following contents:

- o locator - The actual locator, either an IP address or an ASCII string.
- o ifi (integer) - The interface identifier index via which this was discovered - probably no use to a normal ASA
- o expire (system dependent type) - The time on the local system clock when this locator will expire from the cache
- o is_ipaddress (Boolean) - True if the locator is an IP address
- o is_fqdn (Boolean) - True if the locator is an FQDN
- o is_uri (Boolean) - True if the locator is a URI
- o diverted (Boolean) - True if the locator was discovered via a Divert option
- o protocol (integer) - Applicable transport protocol (IPPROTO_TCP or IPPROTO_UDP)
- o port (integer) - Applicable port number

2.3.1.5. Tagged_objective

A 'tagged_objective' parameter is a data structure with the following contents:

- o objective - An objective
- o locator - The ASA_locator associated with the objective, or a null value.

2.3.1.6. Asa_nonce

In most calls, an 'asa_nonce' parameter is required. It is generated when an ASA registers with GRASP, and any call in which an invalid nonce is presented will fail. It is an up to 32-bit opaque value (for example represented as a uint32_t, depending on the language). It should be unpredictable; a possible implementation is to use the same mechanism that GRASP uses to generate Session IDs [I-D.ietf-anima-grasp]. Another possible implementation is to hash the name of the ASA with a locally defined secret key.

2.3.1.7. Session_nonce

In some calls, a 'session_nonce' parameter is required. This is an opaque data structure as far as the ASA is concerned, used to identify calls to the API as belonging to a specific GRASP session (see Section 2.2). In fully threaded implementations this parameter might not be needed, but it is included to act as a session handle if necessary. It will also allow GRASP to detect and ignore malicious calls or calls from timed-out sessions. A possible implementation is to form the nonce from the underlying GRASP Session ID and the source address of the session.

2.3.2. Registration

These functions are used to register an ASA and the objectives that it supports with the GRASP module. If an authorization model is added to GRASP, it would be added here.

o register_asa()

Input parameter:

name of the ASA (UTF-8 string)

Return parameters:

errorcode (integer)

asa_nonce (integer) (if successful)

This initialises state in the GRASP module for the calling entity (the ASA). In the case of success, an 'asa_nonce' is returned which the ASA must present in all subsequent calls. In the case of failure, the ASA has not been authorized and cannot operate.

o deregister_asa()

Input parameters:

asa_nonce (integer)

name of the ASA (UTF-8 string)

Return parameter:

errorcode (integer)

This removes all state in the GRASP module for the calling entity (the ASA), and deregisters any objectives it has registered. Note that these actions must also happen automatically if an ASA crashes.

Note - the ASA name is strictly speaking redundant in this call, but is present for clarity.

o register_objective()

Input parameters:

asa_nonce (integer)
objective (structure)
ttl (integer - default GRASP_DEF_TIMEOUT)
discoverable (Boolean - default False)
overlap (Boolean - default False)
local (Boolean - default False)

Return parameter:

errorcode (integer)

This registers an objective that this ASA supports and may modify. The 'objective' becomes a candidate for discovery. However, discovery responses should not be enabled until the ASA calls listen_negotiate() or listen_synchronize(), showing that it is able to act as a responder. The ASA may negotiate the objective or send synchronization or flood data. Registration is not needed if the ASA only wants to receive synchronization or flood data for the objective concerned.

The 'ttl' parameter is the valid lifetime (time to live) in milliseconds of any discovery response for this objective. The default value should be the GRASP default timeout (GRASP_DEF_TIMEOUT, see [I-D.ietf-anima-grasp]).

If the optional parameter 'discoverable' is True, the objective is immediately discoverable. This is intended for objectives that are only defined for GRASP discovery, and which do not support negotiation or synchronization.

If the optional parameter 'overlap' is True, more than one ASA may register this objective in the same GRASP instance.

If the optional parameter 'local' is True, discovery must return a link-local address. This feature is for objectives that must be restricted to the local link.

This call may be repeated for multiple objectives.

o `deregister_objective()`

Input parameters:

`asa_nonce` (integer)

`objective` (structure)

Return parameter:

`errorcode` (integer)

The 'objective' must have been registered by the calling ASA; if not, this call fails. Otherwise, it removes all state in the GRASP module for the given objective.

2.3.3. Discovery

o `discover()`

Input parameters:

`asa_nonce` (integer)

`objective` (structure)

`timeout` (integer)

`flush` (Boolean - default False)

Return parameters:

`errorcode` (integer)

`locator_list` (structure)

This returns a list of discovered 'ASA_locator's for the given objective. If the optional parameter 'flush' is True, any locally cached locators for the objective are deleted first.

Otherwise, they are returned immediately. If not, GRASP discovery is performed, and all results obtained before the timeout expires are returned. If no results are obtained, an empty list is returned after the timeout. That is not an error condition.

Threaded implementation: This should be called in a separate thread if asynchronous operation is required.

Event loop implementation: An additional read/write 'session_nonce' parameter is used.

2.3.4. Negotiation

- o request_negotiate()

Input parameters:

- asa_nonce (integer)
- objective (structure)
- peer (ASA_locator)
- timeout (integer)

Return parameters:

- errorcode (integer)
- session_nonce (structure) (if successful)
- proffered_objective (structure) (if successful)
- reason (string) (if negotiation declined)

This function opens a negotiation session. The 'objective' parameter must include the requested value, and its loop count should be set to a suitable value by the ASA. If not, the GRASP default will apply.

Note that a given negotiation session may or may not be a dry-run negotiation; the two modes must not be mixed in a single session.

The 'peer' parameter is the target node; it must be an 'ASA_locator' as returned by discover(). If the peer is null, GRASP discovery is performed first.

If the 'errorcode' return parameter is 0, the negotiation has successfully started. There are then two cases:

1. The 'session_nonce' parameter is null. In this case the negotiation has succeeded (the peer has accepted the request). The returned 'proffered_objective' contains the value accepted by the peer.
2. The 'session_nonce' parameter is not null. In this case negotiation must continue. The returned 'proffered_objective' contains the first value proffered by the negotiation peer. Note that this instance of the objective must be used in the subsequent negotiation call because it also contains the current loop count. The 'session_nonce' must be presented in all subsequent negotiation steps.

This function must be followed by calls to 'negotiate_step' and/or 'negotiate_wait' and/or 'end_negotiate' until the negotiation ends. 'request_negotiate' may then be called again to start a new negotiation.

If the 'errorcode' parameter has the value 1 ('declined'), the negotiation has been declined by the peer (M_END and O_DECLINE features of GRASP). The 'reason' string is then available for information and diagnostic use, but it may be a null string. For this and any other error code, an exponential backoff is recommended before any retry.

Threaded implementation: This should be called in a separate thread if asynchronous operation is required.

Event loop implementation: The 'session_nonce' parameter is used in read/write mode.

Special note for the ACP infrastructure ASA: It is likely that this ASA will need to discover and negotiate with its peers in each of its on-link neighbors. It will therefore need to know not only the link-local IP address but also the physical interface and transport port for connecting to each neighbor. One implementation approach to this is to include these details in the 'session_nonce' data structure, which is opaque to normal ASAs.

o listen_negotiate()

Input parameters:

asa_nonce (integer)
objective (structure)

Return parameters:

errorcode (integer)
session_nonce (structure) (if successful)
requested_objective (structure) (if successful)

This function instructs GRASP to listen for negotiation requests for the given 'objective'. It also enables discovery responses for the objective.

Threaded implementation: It will block waiting for an incoming request, so should be called in a separate thread if asynchronous operation is required.

Event loop implementation: A read/write 'session_nonce' parameter is used.

Unless there is an unexpected failure, this call only returns after an incoming negotiation request. When it does so, 'requested_objective' contains the first value requested by the negotiation peer. Note that this instance of the objective must be used in the subsequent negotiation call because it also contains the current loop count. The 'session_nonce' must be presented in all subsequent negotiation steps.

This function must be followed by calls to 'negotiate_step' and/or 'negotiate_wait' and/or 'end_negotiate' until the negotiation ends. 'listen_negotiate' may then be called again to await a new negotiation.

If an ASA is capable of handling multiple negotiations simultaneously, it may call 'listen_negotiate' simultaneously from multiple threads. The API and GRASP implementation must support re-entrant use of the listening state and the negotiation calls. Simultaneous sessions will be distinguished by the threads themselves, the GRASP Session IDs, and the underlying unicast transport sockets.

o stop_listen_negotiate()

Input parameters:

asa_nonce (integer)
objective (structure)

Return parameter:

errorcode (integer)

Instructs GRASP to stop listening for negotiation requests for the given objective, i.e., cancels 'listen_negotiate'.

Threaded implementation: Must be called from a different thread than 'listen_negotiate'.

Event loop implementation: no special considerations.

o negotiate_step()

Input parameters:

asa_nonce (integer)
session_nonce (structure)
objective (structure)
timeout (integer)

Return parameters:

Exactly as for 'request_negotiate'

Executes the next negotiation step with the peer. The 'objective' parameter contains the next value being proffered by the ASA in this step.

Threaded implementation: Called in the same thread as the preceding 'request_negotiate' or 'listen_negotiate', with the same value of 'session_nonce'.

Event loop implementation: Must use the same value of 'session_nonce' returned by the preceding 'request_negotiate' or 'listen_negotiate'.

o negotiate_wait()

Input parameters:

asa_nonce (integer)
session_nonce (structure)
timeout (integer)

Return parameters:

errorcode (integer)

Delay negotiation session by 'timeout' milliseconds.

Threaded implementation: Called in the same thread as the preceding 'request_negotiate' or 'listen_negotiate', with the same value of 'session_nonce'.

Event loop implementation: Must use the same value of 'session_nonce' returned by the preceding 'request_negotiate' or 'listen_negotiate'.

o end_negotiate()

Input parameters:

asa_nonce (integer)
session_nonce (structure)
reply (Boolean)
reason (UTF-8 string)

Return parameters:

errorcode (integer)

End the negotiation session.

'reply' = True for accept (successful negotiation), False for decline (failed negotiation).

'reason' = optional string describing reason for decline.

Threaded implementation: Called in the same thread as the preceding 'request_negotiate' or 'listen_negotiate', with the same value of 'session_nonce'.

Event loop implementation: Must use the same value of 'session_nonce' returned by the preceding 'request_negotiate' or 'listen_negotiate'.

2.3.5. Synchronization and Flooding

- o synchronize()

Input parameters:

- asa_nonce (integer)
- objective (structure)
- peer (ASA_locator)
- timeout (integer)

Return parameters:

- errorcode (integer)
- objective (structure) (if successful)

This call requests the synchronized value of the given 'objective'.

Since this is essentially a read operation, any ASA can do it. Therefore the API checks that the ASA is registered but the objective doesn't need to be registered by the calling ASA.

If the objective was already flooded, the flooded value is returned immediately in the 'result' parameter. In this case, the 'source' and 'timeout' are ignored.

Otherwise, synchronization with a discovered ASA is performed. The 'peer' parameter is an 'ASA_locator' as returned by discover(). If 'peer' is null, GRASP discovery is performed first.

This call should be repeated whenever the latest value is needed.

Threaded implementation: Call in a separate thread if asynchronous operation is required.

Event loop implementation: An additional read/write 'session_nonce' parameter is used.

Since this is essentially a read operation, any ASA can use it. Therefore GRASP checks that the calling ASA is registered but the objective doesn't need to be registered by the calling ASA.

In the case of failure, an exponential backoff is recommended before retrying.

- o `listen_synchronize()`

Input parameters:

- `asa_nonce` (integer)

- `objective` (structure)

Return parameters:

- `errorcode` (integer)

This instructs GRASP to listen for synchronization requests for the given objective, and to respond with the value given in the 'objective' parameter. It also enables discovery responses for the objective.

This call is non-blocking and may be repeated whenever the value changes.

- o `stop_listen_synchronize()`

Input parameters:

- `asa_nonce` (integer)

- `objective` (structure)

Return parameters:

- `errorcode` (integer)

This call instructs GRASP to stop listening for synchronization requests for the given 'objective', i.e. it cancels a previous `listen_synchronize`.

- o `flood()`

Input parameters:

- `asa_nonce` (integer)

ttl (integer)

tagged_objective_list (structure)

Return parameters:

errorcode (integer)

This call instructs GRASP to flood the given synchronization objective(s) and their value(s) and associated locator(s) to all GRASP nodes.

The 'ttl' parameter is the valid lifetime (time to live) of the flooded data in milliseconds (0 = infinity)

The 'tagged_objective_list' parameter is a list of one or more 'tagged_objective' couplets. The 'locator' parameter that tags each objective is normally null but may be a valid 'ASA_locator'. Infrastructure ASAs needing to flood an {address, protocol, port} 3-tuple with an objective create an ASA_locator object to do so. If the IP address in that locator is the unspecified address (':::') it is replaced by the link-local address of the sending node in each copy of the flood multicast, which will be forced to have a loop count of 1. This feature is for objectives that must be restricted to the local link.

The function checks that the ASA registered each objective.

This call may be repeated whenever any value changes.

o get_flood()

Input parameters:

asa_nonce (integer)

objective (structure)

Return parameters:

errorcode (integer)

tagged_objective_list (structure) (if successful)

This call instructs GRASP to return the given synchronization objective if it has been flooded and its lifetime has not expired.

Since this is essentially a read operation, any ASA can do it. Therefore the API checks that the ASA is registered but the objective doesn't need to be registered by the calling ASA.

The 'tagged_objective_list' parameter is a list of 'tagged_objective' couplets, each one being a copy of the flooded objective and a corresponding locator. Thus if the same objective has been flooded by multiple ASAs, the recipient can distinguish the copies.

Note that this call is for advanced ASAs. In a simple case, an ASA can simply call `synchronize()` in order to get a valid flooded objective.

- o `expire_flood()`

Input parameters:

- `asa_nonce` (integer)

- `tagged_objective` (structure)

Return parameters:

- `errorcode` (integer)

This is a call that can only be used after a preceding call to `get_flood()` by an ASA that is capable of deciding that the flooded value is stale or invalid. Use with care.

The 'tagged_objective' parameter is the one to be expired.

2.3.6. Invalid Message Function

- o `send_invalid()`

Input parameters:

- `asa_nonce` (integer)

- `session_nonce` (structure)

- `info` (bytes)

Return parameters:

- `errorcode` (integer)

Sends a GRASP Invalid Message (M_INVALID) message, as described in [I-D.ietf-anima-grasp]. Should not be used if `end_negotiate()` would be sufficient. Note that this message may be used in response to any unicast GRASP message that the receiver cannot interpret correctly. In most cases this message will be generated internally by a GRASP implementation.

'info' = optional diagnostic data. May be raw bytes from the invalid message.

3. Example Logic Flows

TBD

(Until this section is written, some Python examples can be found at <https://github.com/becarpenter/graspy>.)

4. Security Considerations

Security issues for the GRASP protocol are discussed in [I-D.ietf-anima-grasp]. Authorization of ASAs is a subject for future study.

The 'asa_nonce' parameter is used in the API as a first line of defence against a malware process attempting to imitate a legitimately registered ASA. The 'session_nonce' parameter is used in the API as a first line of defence against a malware process attempting to hijack a GRASP session.

5. IANA Considerations

This document does not need IANA assignments.

6. Acknowledgements

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

7. References

7.1. 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.

7.2. Informative References

- [I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-12 (work in progress), October 2017.
- [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-09 (work in progress), October 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-05 (work in progress), October 2017.
- [I-D.liu-anima-grasp-distribution]
Liu, B. and S. Jiang, "Information Distribution over GRASP", draft-liu-anima-grasp-distribution-04 (work in progress), May 2017.
- [RFC7749] Reschke, J., "The "xml2rfc" Version 2 Vocabulary", RFC 7749, DOI 10.17487/RFC7749, February 2016, <<https://www.rfc-editor.org/info/rfc7749>>.

Appendix A. Error Codes

This Appendix lists the error codes defined so far, with suggested symbolic names and corresponding descriptive strings in English. It is expected that complete API implementations will provide for localisation of these descriptive strings.

ok	0	"OK"
declined	1	"Declined"
noReply	2	"No reply"
unspec	3	"Unspecified error"
ASAFull	4	"ASA registry full"
dupASA	5	"Duplicate ASA name"
noASA	6	"ASA not registered"
notYourASA	7	"ASA registered but not by you"
notBoth	8	"Objective cannot support both negotiation and synchronization"
notDry	9	"Dry-run allowed only with negotiation"
notOverlap	10	"Overlap not supported by this implementation"
objFull	11	"Objective registry full"
objReg	12	"Objective already registered"
notYourObj	13	"Objective not registered by this ASA"
notObj	14	"Objective not found"
notNeg	15	"Objective not negotiable"
noSecurity	16	"No security"
noDiscReply	17	"No reply to discovery"
sockErrNegRq	18	"Socket error sending negotiation request"
noSession	19	"No session"
noSocket	20	"No socket"
loopExhausted	21	"Loop count exhausted"
sockErrNegStep	22	"Socket error sending negotiation step"
noPeer	23	"No negotiation peer"
CBORfail	24	"CBOR decode failure"
invalidNeg	25	"Invalid Negotiate message"
invalidEnd	26	"Invalid end message"
noNegReply	27	"No reply to negotiation step"
noValidStep	28	"No valid reply to negotiation step"
sockErrWait	29	"Socket error sending wait message"
sockErrEnd	30	"Socket error sending end message"
IDclash	31	"Incoming request Session ID clash"
notSynch	32	"Not a synchronization objective"
notFloodDisc	33	"Not flooded and no reply to discovery"
sockErrSynRq	34	"Socket error sending synch request"
noListener	35	"No synch listener"
noSynchReply	36	"No reply to synchronization request"
noValidSynch	37	"No valid reply to synchronization request"
invalidLoc	38	"Invalid locator"

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

draft-liu-anima-grasp-api-06, 2017-11-24:

Improved description of event-loop model.

Changed intended status to Informational.

Editorial improvements.

draft-liu-anima-grasp-api-05, 2017-10-02:

Added send_invalid()

draft-liu-anima-grasp-api-04, 2017-06-30:

Noted that simple nodes might not include the API.

Minor clarifications.

draft-liu-anima-grasp-api-03, 2017-02-13:

Changed error return to integers.

Required all implementations to accept objective values in CBOR.

Added non-blocking alternatives.

draft-liu-anima-grasp-api-02, 2016-12-17:

Updated for draft-ietf-anima-grasp-09

draft-liu-anima-grasp-api-02, 2016-09-30:

Added items for draft-ietf-anima-grasp-07

Editorial corrections

draft-liu-anima-grasp-api-01, 2016-06-24:

Updated for draft-ietf-anima-grasp-05

Editorial corrections

draft-liu-anima-grasp-api-00, 2016-04-04:

Initial version

Authors' Addresses

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

Email: brian.e.carpenter@gmail.com

Bing Liu (editor)
Huawei Technologies
Q22, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com

Wendong Wang
BUPT University
Beijing University of Posts & Telecom.
No.10 Xitucheng Road
Hai-Dian District, Beijing 100876
P.R. China

Email: wdwang@bupt.edu.cn

Xiangyang Gong
BUPT University
Beijing University of Posts & Telecom.
No.10 Xitucheng Road
Hai-Dian District, Beijing 100876
P.R. China

Email: xygong@bupt.edu.cn

ANIMA WG
INTERNET-DRAFT
Intended Status: Standard Track
Expires: June 14, 2020

B. Liu
Huawei Technologies
X. Xiao
A. Hecker
MRC, Huawei Technologies
S. Jiang
Huawei Technologies
Z. Despotovic
MRC, Huawei Technologies
December 12, 2019

Information Distribution in Autonomic Networking
draft-liu-anima-grasp-distribution-13

Abstract

This document proposes a solution for information distribution in autonomic networks. Information distribution is categorized into two different modes: 1) instantaneous distribution; 2) publication for retrieval. In the former case, the information is sent, propagates and is disposed of after reception. In the latter case, information needs to be stored in the network.

The capabilities to distribute information are basic and fundamental needs for an autonomous network (cf. ANI [I-D.ietf-anima-reference-model]). This document describes typical use cases of information distribution in ANI and requirements to ANI, such that rich information distribution can be natively supported. The document proposes extensions to the autonomic nodes and suggests an implementation based on GRASP extensions as a protocol on the wire.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at

<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

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

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

Table of Contents

1	Introduction	3
2.	Terminology	3
3.	Requirements of Enriched Information Distribution	4
4.	Node Behaviors	5
4.1	Instant Information Distribution (IID) Sub-module	5
4.2	Asynchronous Information Distribution (AID) Sub-module	6
4.3	Summary	10
5.	Extending GRASP for Information Distribution	10
5.1	Realizing Instant P2P Transmission	10
5.2	Realizing Instant Selective Flooding	11
5.3	Realizing Subscription as An Event	11
5.4	Un_Subscription Objective Option	12
5.5	Publishing Objective Option	12
6.	Security Considerations	13
7.	IANA Considerations	13
8.	References	13
8.1	Normative References	13
8.2	Informative References	13
	Authors' Addresses	14
	Appendix A. Real-world Use Cases of Information Distribution	15
	Appendix B. Information Distribution Module in ANI	18
	Appendix C. Asynchronous Information Distribution Integrated with GRASP APIs	18

1 Introduction

In an autonomic network, autonomic functions (AFs) running on autonomic nodes constantly exchange information, e.g. AF control/management signaling or AF data exchange. This document discusses the information distribution capability of such exchanges between AFs.

Depending on the number of participants, the information can be distributed in in the following scenarios:

- 1) Point-to-point (P2P) Communication: information is exchanged between parties, i.e. two nodes.
- 2) One-to-Many Communication: information exchanges involve an information source and multiple receivers.

The approaches to information distribution can be chiefly categorized into two basic modes:

- 1) An instantaneous mode (push): a source sends the actual content (e.g. control/management signaling, synchronization data and so on) to all interested receiver(s) immediately. Generally, some preconfiguration is required, as nodes interested in this information must be already known to all nodes in the sense that any receiving node must be able to decide, to which nodes this data is to be sent.
- 2) An asynchronous mode (delayed pull): here, a source publishes the content in some form in the network, which may later be looked for, found and retrieved by some endpoints in the AN. Here, depending on the size of the content, either the whole content or only its metadata might be published into the AN. In the latter case the metadata (e.g. a content descriptor, e.g. a key, and a location in the ANI) may be used for the actual retrieval. Importantly, the source, i.e. here publisher, needs to be able to determine the node, where the information (or its metadata) can be stored.

To avoid repetitive implementations by each AF developer, this document opts for a common support for information distribution implemented as a basic ANI capability, therefore available to all AFs. In fact, GRASP already provides part of the capabilities.

Regardless, an AF may still define and implement its own information distribution capability. Such a capability may then be advertised using the common information distribution capability defined in this document. Overall, ANI nodes and AFs may decide, which of the

information distribution mechanisms they want to use for which type of information, according to their own preferences (e.g. semantic routing table, etc.)

This document first analyzes requirements for information distribution in autonomic networks (Section 3) and then discuss the relevant node behavior (Section 4). After that, the required GRASP extensions are formally introduced (Section 5).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Requirements for Information Distribution in ANI

The question of information distribution in an autonomic network can be discussed through particular use cases or more generally. Depending on the situation it can be quite simple or might require more complex provisions.

Indeed, in the simplest case, the information can be sent:

- 1) at once (in one packet, in one flow),
- 2) straightaway (send-and-forget),
- 3) to all nodes.

Presuming 1), 2) and 3) hold, information distribution in smaller or scarce topologies can be implemented using broadcast, i.e. unconstrained flooding. For reasons well-understood, this approach has its limits in larger and denser networks. In this case, a graph can be constructed such that it contains every node exactly once (e.g. a spanning tree), still allowing to distribute any information to all nodes straightaway. Multicast tree construction protocols could be used in this case. There are reasonable use cases for such scenarios, as presented in Appendix B.

A more complex scenario arises, if only 1) and 2) hold, but the information only concerns a subset of nodes. Then, some kind of selection becomes required, to which nodes the given information should be distributed. Here, a further distinction is necessary; notably, if the selection of the target nodes is with respect to the nature or position of the node, or whether it is with respect to the information content. If the first, some knowledge about the node types, its topological position, etc (e.g. the routing information within ANI) can be used to distinguish nodes accordingly. For instance, edge nodes and forwarding nodes can be distinguished in this way. If the distribution scope is primarily to be defined by the information elements, then a registration / join / subscription or label distribution mechanism is unavoidable. This would be the case, for instance, if the AFs can be dynamically deployed on nodes, and the information is majorily destined to the AFs. Then, depending on the current AF deployment, the distribution scope must be adjusted as well.

If only 1) holds, but the information content might be required again and again, or might not yet be fully available, then more complex mechanisms might be required to store the information within the network for later, for further redistribution, and for notification of interested nodes. Examples for this include distribution of reconfiguration information for different AF instances, which might not require an immediate action, but only an eventual update of the parameters. Also, in some situations, there could be a significant

delay between the occurrence of a new event and the full content availability (e.g. if the processing requires a lot of time).

Finally, none of the three might hold. Then, along with the subscription and notification, the actual content might be different from its metadata, i.e. some description of the content and, possibly, its location. The fetching can then be implemented in different, appropriate ways, if necessary as a complex transport session.

In essence, as flooding is usually not an option, and the interest of nodes for particular information elements can change over time, ANI should support autonomies also for the information distribution.

This calls for autonomic mechanisms in the ANI, allowing participating nodes to 1) advertise or publish 2) look for or subscribe to 3) store 4) fetch/retrieve 5) instantaneously push information elements.

In the following cases, situations depicting diverse information distribution needs are discussed.

1) Long Communication Intervals. The actual sending of the information is not necessarily instantaneous with some event. Advanced AFs may involve into longer jobs/tasks (e.g. database lookup, authentication etc.) when processing requests, and might not be able to reply immediately. Instead of actively waiting for the reply, a better way for an interested AF might be to get notified, when the reply is finally available.

2) Common Interest Distribution. AFs may share interest in common information. For example, the network intent will be distributed to network nodes enrolled, which is usually one-to-many scenario. Intent distribution can also be performed by an instant flooding (e.g. via GRASP) to every network node. However, because of network dynamics, not every node can be just ready at the moment when the network intent is broadcast. Also, a flooding often does not cover all network nodes as there is usually a limitation on the hop number. In fact, nodes may join in the network sequentially. In this situation, an asynchronous communication model could be a better choice where every (newly joining) node can subscribe the intent information and will get notified if it is ready (or updated).

3) Distributed Coordination. With computing and storage resources on autonomic nodes, alive AFs not only consume but also generate data information. An example is AFs coordinating with each other as distributed schedulers, responding to service requests and

distributing tasks. It is critical for those AFs to make correct decisions based on local information, which might be asymmetric as well. AFs may also need synthetic/aggregated data information (e.g. statistic info, like average values of several AFs, etc.) to make decisions. In these situations, AFs will need an efficient way to form a global view of the network (e.g. about resource consumption, bandwidth and statistics). Obviously, purely relying on instant communication model is inefficient, while a scalable, common, yet distributed data layer, on which AFs can store and share information in an asynchronous way, should be a better choice.

Therefore, for ANI, in order to support various communication scenarios, an information distribution module is required, and both instantaneous and asynchronous communication models should be supported. Some real-world use cases are introduced in Appendix A.

4. Node Behaviors

In this section, how a node should behave in order to support the two identified modes of information distribution is discussed. An ANI is a distributed system, so the information distribution module must be implemented in a distributed way as well.

4.1 Instant Information Distribution (IID) Sub-module

In this case, An information sender directly specifies the information receiver(s). The instant information distribution sub-module will be the main element.

4.1.1 Instant P2P Communication

IID sub-module performs instant information transmission for ASAs running in an ANI. In specific, IID sub-module will have to retrieve the address of the information receiver specified by an ASA, then deliver the information to the receiver. Such a delivery can be done either in a connectionless or a connection-oriented way.

Current GRASP provides the capability to support instant P2P synchronization for ASAs. A P2P synchronization is a use case of P2P information transmission. However, as mention in Section 3, there are some scenarios where one node needs to transmit some information to another node(s). This is different to synchronization because after transmitting the information, the local status of the information does not have to be the same as the information sent to the receiver. This is not directly support by existing GRASP.

4.1.2 Instant Flooding Communication

IID sub-module finishes instant flooding for ASAs in an ANI. Instant flooding is for all ASAs in an ANI. An information sender has to specify a special destination address of the information and broadcast to all interfaces to its neighbors. When another IID sub-module receives such a broadcast, after checking its TTL, it further broadcast the message to the neighbors. In order to avoid flooding storms in an ANI, usually a TTL number is specified, so that after a pre-defined limit, the flooding message will not be further broadcast again.

In order to avoid unnecessary flooding, a selective flooding can be done where an information sender wants to send information to multiple receivers at once. When doing this, sending information needs to contain criteria to judge on which interfaces the distributed information should and should not be sent. Specifically, the criteria contain:

- o Matching Condition: a set of matching rules such as addresses of recipients, node features and so on.
- o Action: what the node needs to do when the Matching Condition is fulfilled. For example, the action could be forwarding or discarding the distributed message.

Sent information must be included in the message distributed from the sender. The receiving node reacts by first checking the carried Matching Condition in the message to decide who should consume the message, which could be either the node itself, some neighbors or both. If the node itself is a recipient, Action field is followed; if a neighbor is a recipient, the message is sent accordingly.

An exemplary extension to support selective flooding on GRASP is described in Section 5.

4.2 Asynchronous Information Distribution (AID) Sub-module

In asynchronous information distribution, sender(s) and receiver(s) are not immediately specified while they may appear in an asynchronous way. Firstly, AID sub-module enables that the information can be stored in the network; secondly, AID sub-module provides an information publication and subscription (Pub/Sub) mechanism for ASAs.

As sketched in the previous section, in general each node requires two modules: 1) Information Storage (IS) module and 2) Event Queue (EQ) module in the information distribution module. Details of the

two modules are described in the following sections.

4.2.1 Information Storage

IS module handles how to save and retrieve information for ASAs across the network. The IS module uses a syntax to index information, generating the hash index value (e.g. a hash value) of the information and mapping the hash index to a certain node in ANI. Note that, this mechanism can use existing solutions. Specifically, storing information in an ANIMA network will be realized in the following steps.

- 1) ASA-to-IS Negotiation. An ASA calls the API provided by information distribution module (directly supported by IS sub-module) to request to store the information somewhere in the network. The IS module performs various checks of the request (e.g. permitted information size).
- 2) Storing Peer Mapping. The information block will be handled by the IS module in order to calculate/map to a peer node in the network. Since ANIMA network is a peer-to-peer network, a typical way is to use distributed hash table (DHT) to map information to a unique index identifier. For example, if the size of the information is reasonable, the information block itself can be hashed, otherwise, some meta-data of the information block can be used to generate the mapping.
- 3) Storing Peer Negotiation Request. Negotiation request of storing the information will be sent from the IS module to the IS module on the destination node. The negotiation request contains parameters about the information block from the source IS module. According to the parameters as well as the local available resource, the requested storing peer will send feedback the source IS module.
- 4) Storing Peer Negotiation Response. Negotiation response from the storing peer is sent back to the source IS module. If the source IS module gets confirmation that the information can be stored, source IS module will prepare to transfer the information block; otherwise, a new storing peer must be discovered (i.e. going to step 7).
- 5) Information Block Transfer. Before sending the information block to the storing peer that already accepts the request, the IS module of the source node will check if the information block can be afforded by one GRASP message. If so, the information block will be directly sent by calling a GRASP API. Otherwise, a bulk data transmission is needed. For that, there are multiple ways to

do it.

The first option is to utilize one of existing protocols that is independent of the GRASP stack. For example, a session connectivity can be established to the storing peer, and over the connection the bulky data can be transmitted part by part. In this case, the IS module should support basic TCP-based session protocols such as HTTP(s) or native TCP.

The second option is to directly use GRASP itself for bulky data transferring. [I-D.carpenter-anima-grasp-bulk-04].

6) Information Writing. Once the information block (or a smaller block) is received, the IS module of the storing peer will store the data block in the local storage is accessible.

7) (Optional) New Storing Peer Discovery. If the previously selected storing peer is not available to store the information block, the source IS module will have to identify a new destination node to start a new negotiation. In this case, the discovery can be done by using discovery GRASP API to identify a new candidate, or more complex mechanisms can be introduced.

Similarly, Getting information from an ANI will be realized in the following steps.

1) ASA-to-IS Request. An ASA accesses the IS module via the APIs exposed by the information distribution module. The key/index of the interested information will be sent to the IS module. An assumption here is that the key/index should be known to an ASA before an ASA can ask for the information. This relates to the publishing/subscribing of the information, which are handled by other modules (e.g. Event Queue with Pub/Sub supported by GRASP).

2) Storing Peer Mapping. IS module maps the key/index of the requested information to a peer that stores the information, and prepares the information request. The mapping here follows the same mechanism when the information is stored.

3) Retrieval Negotiation Request. The source IS module sends a request to the storing peer and asks if such an information object is available.

4) Retrieval Negotiation Response. The storing peer checks the key/index of the information in the request, and replies to the source IS module. If the information is found and the information block can be afforded within one GRASP message, the information will be sent together with the response to the source IS module.

5) (Optional) New Destination Request. If the information is not found after the source IS module gets the response from the originally identified storing peer, the source IS module will have to discover the location of the requested information.

IS module can reuse distributed databases and key value stores like NoSQL, Cassandra, DHT technologies. storage and retrieval of information are all event-driven responsible by the EQ module.

4.2.2 Event Queue The Event Queue (EQ) module is to help ASAs to publish information to the network and subscribe to interested information in asynchronous scenarios. In an ANI, information generated on network nodes is an event labeled with an event ID, which is semantically related to the topic of the information. Key features of EQ module are summarized as follows.

1) Event Group: An EQ module provides isolated queues for different event groups. If two groups of AFs could have completely different purposes, the EQ module allows to create multiple queues where only AFs interested in the same topic will be aware of the corresponding event queue.

2) Event Prioritization: Events can have different priorities in ANI. This corresponds to how much important or urgent the event implies. Some of them are more urgent than regular ones. Prioritization allows AFs to differentiate events (i.e. information) they publish or subscribe to.

3) Event Matching: an information consumer has to be identified from the queue in order to deliver the information from the provider. Event matching keeps looking for the subscriptions in the queue to see if there is an exact published event there. Whenever a match is found, it will notify the upper layer to inform the corresponding ASAs who are the information provider and subscriber(s) respectively.

The EQ module on every network node operates as follows.

1) Event ID Generation: If information of an ASA is ready, an event ID is generated according to the content of the information. This is also related to how the information is stored/saved by the IS module introduced before. Meanwhile, the type of the event is also specified where it can be of control purpose or user plane data.

2) Priority Specification: According to the type of the event, the ASA may specify its priority to say how this event is to be processed. By considering both aspects, the priority of the event will be determined.

3) Event Enqueue: Given the event ID, event group and its priority, a queue is identified locally if all criteria can be satisfied. If there is such a queue, the event will be simply added into the queue, otherwise a new queue will be created to accommodate such an event.

4) Event Propagation: The published event will be propagated to the other network nodes in the ANIMA domain. A propagation algorithm can be employed to optimize the propagation efficiency of the updated event queue states.

5) Event Match and Notification: While propagating updated event states, EQ module in parallel keeps matching published events and its interested consumers. Once a match is found, the provider and subscriber(s) will be notified for final information retrieval.

The category of event priority is defined as the following. In general, there are two event types:

1) Network Control Event: This type of events are defined by the ANI for operational purposes on network control. A pre-defined priority levels for required system messages is suggested. For highest level to lowest level, the priority value ranges from NC_PRIOR_HIGH to NC_PRIOR_LOW as integer values. The NC_PRIOR_* values will be defined later according to the total number system events required by the ANI;

2) Custom ASA Event: This type of events are defined by the ASAs of users. This specifies the priority of the message within a group of ASAs, therefore it is only effective among ASAs that join the same message group. Within the message group, a group header/leader has to define a list of priority levels ranging from CUST_PRIOR_HIGH to CUST_PRIOR_LOW. Such a definition completely depends on the individual purposes of the message group.

When a system message is delivered, its event type and event priority value have to be both specified;

Event contains the address where the information is stored, after a subscriber is notified, it directly retrieves the information from the given location.

4.3 Summary

In summary, the general requirements for the information distribution module on each autonomic node are realized by two sub-modules handling instant communications and asynchronous communications, respectively. For instantaneous mode, node requirements are simple,

calling for support for additional signaling. With minimum efforts, reusing the existing GRASP is possible.

For asynchronous mode, information distribution module uses new primitives on the wire, and implements an event queue and an information storage mechanism. An architectural consideration on ANI with the information distribution module is briefly discussed in Appendix B.

5. Extending GRASP for Information Distribution

5.1 Realizing Instant P2P Transmission

This could be a new message in GRASP. In fragmentary CDDL, an Un-solicited Synchronization message follows the pattern:

```
unsolicited_synch-message = [M_UNOLSOLIDSYNCH, session-id,
                             objective]
```

A node MAY actively send a unicast Un-solicited Synchronization message with the Synchronization data, to another node. This MAY be sent to port GRASP_LISTEN_PORT at the destination address, which might be obtained by GRASP Discovery or other possible ways. The synchronization data are in the form of GRASP Option(s) for specific synchronization objective(s).

5.2 Realizing Instant Selective Flooding

Since normal flooding is already supported by GRASP, this section only defines the selective flooding extension.

In fragmentary CDDL, the selective flooding follows the pattern:

```
selective-flood-option = [O_SELECTIVE_FLOOD, +O_MATCH-CONDITION,
                          match-object, action]
```

```
O_MATCH-CONDITION = [O_MATCH-CONDITION, Obj1, match-rule, Obj2]
Obj1 = text
```

```
match-rule = GREATER / LESS / WITHIN / CONTAIN
```

```
Obj2 = text
```

```
match-object = NEIGHBOR / SELF
```

```
action = FORWARD / DROP
```

The option field encapsulates a match-condition option which represents the conditions regarding to continue or discontinue flood the current message. For the match-condition option, the Obj1 and Obj2 are to objects that need to be compared. For example, the Obj1 could be the role of the device and Obj2 could be "RSG". The match rules between the two objects could be greater, less than, within, or contain. The match-object represents of which Obj1 belongs to, it could be the device itself or the neighbor(s) intended to be flooded. The action means, when the match rule applies, the current device just continues flood or discontinues.

5.3 Realizing Subscription as An Event

In fragmentary CDDL, a Subscription Objective Option follows the pattern:

```
subscription-objection-option = [SUBSCRIPTION, 2, 2, subobj]
objective-name = SUBSCRIPTION

objective-flags = 2

loop-count = 2

subobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a subscription to a specific object.

5.4 Un_Subscription Objective Option

In fragmentary CDDL, a Un_Subscribe Objective Option follows the pattern:

```
Unsubscribe-objection-option = [UNSUBSCRIB, 2, 2, unsubobj]
objective-name = SUBSCRIPTION
objective-flags = 2
loop-count = 2
unsubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a un-subscription to a specific object.

5.5 Publishing Objective Option

In fragmentary CDDL, a Publish Objective Option follows the pattern:

```
publish-objection-option = [PUBLISH, 2, 2, pubobj] objective-name
= PUBLISH
objective-flags = 2
loop-count = 2
pubobj = text
```

This option MAY be included in GRASP M_Synchronization, when included, it means this message is for a publish of a specific object data.

6. Security Considerations

The distribution source authentication could be done at multiple layers:

- o Outer layer authentication: the GRASP communication is within ACP (Autonomic Control Plane, [I-D.ietf-anima-autonomic-control-plane]). This is the default GRASP behavior.
- o Inner layer authentication: the GRASP communication might not be within a protected channel, then there should be embedded protection in distribution information itself. Public key infrastructure might be involved in this case.

7. IANA Considerations

TBD.

8. References

8.1 Normative References

```
[I-D.ietf-anima-grasp]
    Bormann, D., Carpenter, B., and B. Liu, "A Generic
    Autonomic Signaling Protocol (GRASP)", draft-ietf-
    animagrasp-15 (Standard Track), October 2017.
```

8.2 Informative References

```
[RFC7575] Behringer, M., "Autonomic Networking: Definitions and
    Design Goals", RFC 7575, June 2015
```

```
[I-D.ietf-anima-autonomic-control-plane]
    Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic
    Control Plane (ACP)", draft-behringer-anima-autonomic-
    control-plane-13, December 2017.
```

```
[I-D.ietf-anima-stable-connectivity-10]
```

Eckert, T., Behringer, M., "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-ietf-anima-stable-connectivity-10, February 2018.

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

Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Pierre P., Liu, B., Nobre, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-05, October 2017.

[I-D.du-anima-an-intent]

Du, Z., Jiang, S., Nobre, J., Ciavaglia, L., and M. Behringer, "ANIMA Intent Policy and Format", draft-duanima-an-intent-05 (work in progress), February 2017.

[I-D.ietf-anima-grasp-api]

Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", draft-ietf-anima-grasp-api-00 (work in progress), December 2017.

[I-D.carpenter-anima-grasp-bulk-04]

Carpenter, B., Jiang, S., Liu, B., "Transferring Bulk Data over the GeneRIC Autonomic Signaling Protocol (GRASP)", draft-carpenter-anima-grasp-bulk-04 (work in progress), July 3, 2019

[3GPP.29.500]

3GPP, "Technical Realization of Service Based Architecture", 3GPP TS 29.500 15.1.0, 09 2018

[3GPP.23.501]

3GPP, "System Architecture for the 5G System", 3GPP TS 23.501 15.2.0, 6 2018, <<http://www.3gpp.org/ftp/Specs/html-info/23501.htm>>.

[3GPP.23.502]

3GPP, "Procedures for the 5G System", 3GPP TS 23.502 15.2.0, 6 2018, <<http://www.3gpp.org/ftp/Specs/html-info/23502.htm>>.

[5GAA.use.cases]

White Paper "Toward fully connected vehicles: Edge computing for advanced automotive communications", 5GAA <<http://5gaa.org/news/toward-fully-connected-vehicles-edge-computing-for-advanced-automotive-communications/>>

Authors' Addresses

Bing Liu
Huawei Technologies
Q27, Huawei Campus

No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com

Sheng Jiang
Huawei Technologies
Q27, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: jiangsheng@huawei.com

Xun Xiao
Munich Research Center
Huawei technologies
Riesstr. 25, 80992, Muenchen, Germany

Emails: xun.xiao@huawei.com

Artur Hecker
Munich Research Center
Huawei technologies
Riesstr. 25, 80992, Muenchen, Germany

Emails: artur.hecker@huawei.com

Zoran Despotovic
Munich Research Center
Huawei technologies
Riesstr. 25, 80992, Muenchen, Germany

Emails: zoran.despotovic@huawei.com

Appendix A. Real-world Use Cases of Information Distribution

The requirement analysis in Section 3 shows that generally information distribution should be better 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 5G mobile networks from 3GPP has been defined to follow a service-based architecture (SBA) where any network function (NF) can be dynamically associated with any other NF(s) when needed to compose a network service. Note that one NF can simultaneously associate with multiple other NFs, instead of being physically wired as in the previous generations of mobile networks. NFs communicate with each other over service-based interface (SBI), which is also standardized by 3GPP [3GPP.23.501].

In order to realize an SBA network system, detailed requirements are further defined to specify how NFs should interact with each other with information exchange over the SBI. We now list three requirements that are related to information distribution here.

- 1) NF Pub/Sub: Any NF should be able to expose its service status to the network and any NF should be able to subscribe the service status of an NF and get notified if the status is available. A concrete example is that a session management function (SMF) can subscribe to the REGISTER notification from an access management function (AMF) if there is a new user equipment trying to access the mobile network [3GPP.23.502].

- 2) Network Exposure Function (NEF): A particular network function that is required to manage the event exposure and distributions. Specifically, SBA requires such a functionality to register network events from the other NFs (e.g. AMF, SMF and so on), classify the events and properly handle event distributions accordingly in terms of different criteria (e.g. priorities) [3GPP.23.502].

- 3) Network Repository Function (NRF): A particular network function where all service status information is stored for the whole network. An SBA network system requires all NFs to be stateless so as to improve the resilience as well as agility of providing network services. Therefore, the information of the available NFs and the service status generated by those NFs will

be globally stored in NRF as a repository of the system. This clearly implies storage capability that keeps the information in the network and provides those information when needed. A concrete example is that whenever a new NF comes up, it first of all registers itself at NRF with its profile. When a network service requires a certain NF, it first inquires NRF to retrieve the availability information and decides whether or not there is an available NF or a new NF must be instantiated [3GPP.23.502].

(Note: 3GPP CT adopted HTTP2.0/JSON to be the protocol communicating between NFs, but autonomic networks can also load HTTP2.0 with in ACP.)

A.2 Vehicle-to-Everything

Connected car is one of scenarios interested in automotive manufacturers, carriers and vendors. 5G Automotive Alliance - an industry collaboration organization defines many promising use cases where services from car industry should be supported by the 5G mobile network. Here we list two examples as follows [5GAA.use.cases].

1) Software/Firmware Update: Car manufacturers expect that the software/firmware of their car products can be remotely updated/upgraded via 5G network, instead of onsite visiting their 4S stores/dealers offline as nowadays. This requires the network to provide a mechanism for vehicles to receive the latest software updates during a certain period of time. In order to run such a service for a car manufacturer, the network shall not be just like a network pipe anymore. Instead, information data have to be stored in the network, and delivered in a publishing/subscribing fashion. For example, the latest release of a software will be first distributed and stored at the access edges of the mobile network, after that, the updates can be pushed by the car manufacturer or pulled by the car owner as needed.

2) Real-time HD Maps: Autonomous driving clearly requires much finer details of road maps. Finer details not only include the details of just static road and streets, but also real-time information on the road as well as the driving area for both local urgent situations and intelligent driving scheduling. This asks for situational awareness at critical road segments in cases of changing road conditions. Clearly, a huge amount of traffic data that are real-time collected will have to be stored and shared across the network. This clearly requires the storage capability, data synchronization and event notifications in urgent cases from the network, which are still missing at the infrastructure layer.

A.3 Summary

Through the general analysis and the concrete examples from the real-world, we realize that the ways information are exchanged in the coming new scenarios are not just short and instant anymore. More advanced as well as diverse information distribution capabilities are required and should be generically supported from the infrastructure layer. Upper layer applications (e.g. ASAs in ANIMA) access and utilize such a unified mechanism for their own services.

Appendix B. Information Distribution Module in ANI

This section describes how the information distribution module fits into the ANI and what extensions of GRASP are required [I-D.ietf-anima-grasp].

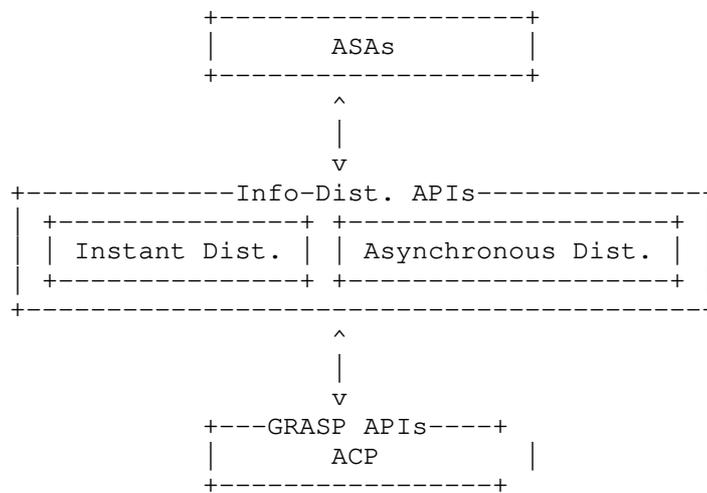


Figure 1. Information Distribution Module and GRASP Extension.

As the Fig 1 shows, the information distribution module two sub-modules for instant and asynchronous information distributions, respectively, and provides APIs to ASAs. Specific Behaviors of modules are described in Section 5.

Appendix C. Asynchronous ID Integrated with GRASP APIs

Actions triggered to the information distribution module will eventually invoke underlying GRASP APIs. Moreover, EQ and IS modules are usually correlated. When an AF(ASA) publishes information, not only such an event is translated and sent to EQ module, but also the information is indexed and stored simultaneously. Similarly, when an AF(ASA) subscribes information, not only subscribing event is

triggered and sent to EQ module, but also the information will be retrieved by IS module at the same time.

o Storing and publishing information: This action involves both IS and EQ modules where a node that can store the information will be discovered first and related event will be published to the network. For this, GRASP APIs `discover()`, `synchronize()` and `flood()` are combined to compose such a procedure. In specific, `discover()` call will 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.

ANIMA
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

P. Peloso
L. Ciavaglia
Nokia
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].

Status of This Memo

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

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

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

This Internet-Draft will expire on September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Problem statement	3
2. Motivations from an operator viewpoint	4
2.1. Illustration of increasingly constraining operator's objectives	4
2.2. Deployment scenarios of autonomic functions	5
2.3. Operator's requirements with regards to autonomic functions	9
3. Installation phase	10
3.1. Operator's goal	10
3.2. Installation phase inputs and outputs	11
4. Instantiation phase	12
4.1. Operator's goal	12
4.2. Instantiation phase inputs and outputs	13
4.3. Instantiation phase requirements	13
5. Operation phase	14
6. Autonomic Function Agent specifications	15
6.1. Life-cycle	15
6.2. ASA Class Manifest	16
6.3. ASA Instance Mandate	17
6.4. ASA Instance Manifest	18
7. Implication for other ANIMA components	19
7.1. Additional entities for ANIMA ecosystem	19
7.2. Requirements for GRASP and ACP messages	20
7.2.1. Control when an ASA runs	21
7.2.2. Know what an ASA does to the network	21
7.2.3. Decide which ASA control which equipment	22
8. Acknowledgments	22
9. IANA Considerations	22
10. Security Considerations	22
11. References	22

11.1. Normative References 22
 11.2. Informative References 23
 Authors' Addresses 23

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 60%.

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:

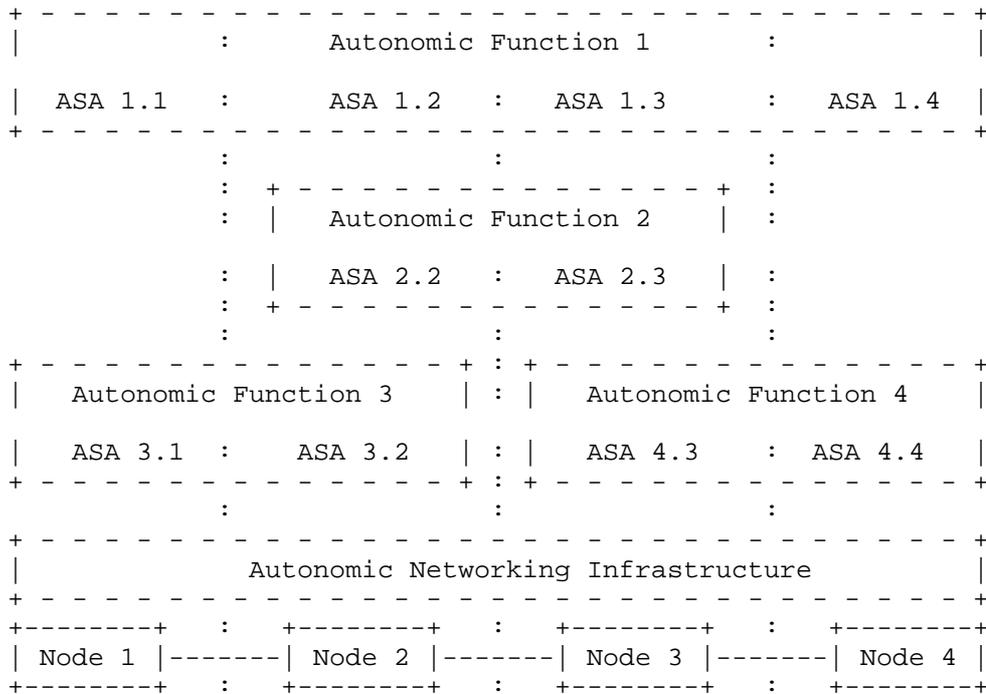


Figure 1: High level view of an Autonomic Network

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.

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.

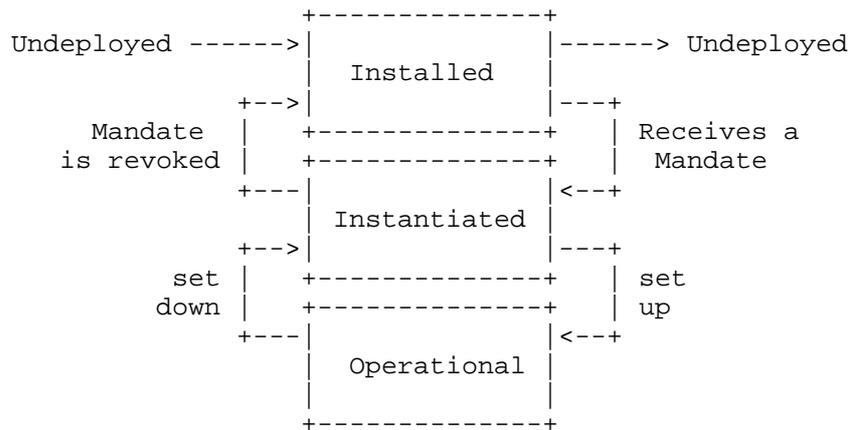


Figure 3: Life cycle of an Autonomic Function Agent

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

This 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