Stream: Internet Engineering Task Force (IETF)
RFC: 8994
Category: Standards Track
Published: May 2021
ISSN: 2070-1721
Authors: T. Eckert, Ed. *Futurewei USA*
M. Behringer, Ed.
S. Bjarnason *Arbor Networks*

RFC 8994
An Autonomic Control Plane (ACP)

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 be 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 is misconfigured.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8994.

Copyright Notice

Table of Contents

1. Introduction (Informative)

Autonomic Networking is a concept of self-management: autonomic functions self-configure, and negotiate parameters and settings across the network. "Autonomic Networking: Definitions and Design Goals" [RFC7575] defines the fundamental ideas and design goals of Autonomic Networking. A gap analysis of Autonomic Networking is given in "General Gap Analysis for Autonomic Networking" [RFC7576]. The reference architecture for Autonomic Networking in the IETF is specified in the document "A Reference Model for Autonomic Networking" [RFC8993].

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

Today, the OAM and control plane of IP networks is what is typically called in-band management and/or 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 dependencies, can disrupt OAM and control operations and especially disrupt remote management access to the affected node itself and potentially disrupt access to a much larger number of nodes for which the affected node is on the network path.

For an example of in-band management failing in the face of operator-induced misconfiguration, see [FCC], for example, Section 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. In the worst-case scenario, personnel are sent on site to access devices through out-of-band management ports (also called craft ports, serial consoles, or management Ethernet ports). However, both options are expensive.

In increasingly automated networks, both centralized management systems and distributed autonomic service agents in the network require a control plane that 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 of how this is achieved are described in Section 6.

In a fully Autonomic Network without legacy control or management functions and/or protocols, the data plane would be just a forwarding plane for "data" IPv6 packets, which are packets other than those control and management plane packets forwarded by the ACP itself. In such a network, there would be no non-autonomous control of nodes nor a non-autonomous management plane.

Routing protocols would be built inside the ACP as autonomous functions via autonomous service agents, leveraging the following ACP 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 session and presentation functions of the control and management protocol.

When ACP functionality is added to nodes that do not have 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, preexisting, non-autonomous management and/or control plane is considered to be part of the data plane to avoid introducing more complex terminology only for this case.

Like the forwarding plane for "data" packets, the non-autonomous control and management plane functions can then be managed and/or used via the ACP. This terminology is consistent with preexisting documents such as "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)" [RFC8368].

In both autonomous and non-autonomous instances, the ACP is built such that it operates in the absence of the data plane. The ACP also operates in the presence of any (mis)configured non-autonomous management and/or control components in the data plane.

The ACP serves several purposes simultaneously:

1. Autonomic functions communicate over the ACP. The ACP therefore directly supports Autonomic Networking functions, as described in [RFC8993]. For example, GRASP ("GeneRic Autonomic Signaling Protocol (GRASP)" [RFC8990]) 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 ACP to securely bootstrap network devices in remote locations, even if the (data plane) network in between is not yet configured; no bootstrap configuration that is dependent on the data plane is required. An example of such a secure bootstrap process is described in "Bootstrapping Remote Secure Key Infrastructure (BRSKI)" [RFC8995].
3. An operator can use ACP to access remote devices using protocols such as Secure SHell (SSH) or Network Configuration Protocol (NETCONF), even if the network is misconfigured or unconfigured.

This document describes these purposes as use cases for the ACP in Section 3, and 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 Layer 2 (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 the benefits of the ACP (after all the details have been defined). Section 9 provides operational recommendations. Appendix A provides additional background and describes possible extensions that were not applicable for this specification but were considered important to document. There are no dependencies on Appendix A in order to build a complete working and interoperable ACP according to this document.

The ACP provides secure IPv6 connectivity; therefore, it can be used for secure connectivity not only for self-management as required for the ACP in [RFC7575] but also for traditional (centralized) management. The ACP can be implemented and operated without any other

components of Autonomic Networks, except for GRASP. ACP relies on per-link Discovery Unsolicited Link-Local (DULL) GRASP (see Section 6.4) to auto-discover 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 [RFC8368] describes how the ACP can be used alone to provide secure and stable connectivity for autonomic and non-autonomic OAM applications, specifically for the case of current non-autonomic networks and/or 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 Infrastructure (BRSKI) (see [RFC8995]) results in the "Autonomic Network Infrastructure" (ANI) as defined in [RFC8993], 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 automation in SDN style (see "Software-Defined Networking (SDN): Layers and Architecture Terminology" [RFC7426]) or distributed automation via Autonomic Service Agents (ASA) and/or Autonomic Functions (AF), or a mixture of both. See [RFC8993] for more information.

1.1. Applicability and Scope

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

The design of the ACP as defined in this document is considered to be applicable to all types of "professionally managed" networks: Service Provider, Local Area Network (LAN), Metropolitan Area Network (MAN/Metro), Wide Area Network (WAN), Enterprise Information Technology (IT) and Operational Technology (OT) networks. The ACP can operate equally on Layer 3 (L3) equipment and on L2 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 "Security Architecture for the Internet Protocol" [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 EST (see "Enrollment over Secure Transport" [RFC7030]), GRASP, UDP, TCP, and Transport Layer Security (TLS, see Section 6.1). For more information regarding the 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 RPL, see "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks" [RFC6550], which is separate from routing and forwarding for the data plane (user traffic).

The ACP uses only IPv6 to avoid the complexity of dual-stack (both IPv6 and IPv4) ACP operations. Nevertheless, it can be integrated without any changes to 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, IPv6 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 protocol suitable for providing 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 and/or variations of the ACP could better meet expectations that are different from those upon which the current design is based, including supporting constrained devices better.

2. Acronyms and Terminology (Informative)

This document serves both as a normative specification for ACP node behavior as well as an explanation of the context by providing descriptions of requirements, benefits, architecture, and operational aspects. Normative sections are labeled "(Normative)" and use BCP 14 keywords. Other sections are labeled "(Informative)" and do not use those normative keywords.

In the rest of the document, we will refer to systems that use 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). Introduced terms 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 or set: The ACP address may imply a range or set of addresses that the node can assign for different purposes. This address range or set is derived by the node from the format of the ACP address called the addressing sub-scheme.

ACP certificate: A Local Device IDentity (LDevID) certificate conforming to "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile" [RFC5280] that carries 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. In the context of the ANI, the ACP certificate is also called the ANI certificate. In the context of AN, the ACP certificate is also called the AN certificate.

ACP connect interface: An interface on an ACP node that provides 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 loopback interface: The loopback interface in the ACP VRF that has the ACP address assigned to it. See Section 6.13.5.1.

ACP network: The ACP network comprises 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 or simple case, the ACP has one Unique Local Address (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 channels are established between (normally) adjacent ACP nodes to carry ACP VRF traffic in-band over the same links and paths as data plane traffic but isolate it from the data plane traffic and secure it.

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 DTLS.

ACP virtual interface: An interface in the ACP VRF mapped to one or more ACP secure channels. See Section 6.13.5.

acp-node-name field: An information field in the ACP certificate in which the following ACP-relevant information is encoded: the ACP domain name, the ACP IPv6 address of the node, and optional additional role attributes about the node.

AN: Autonomic Network. A network according to [RFC8993]. 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, and GRASP) are also described as ASAs.

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

BRSKI: Bootstrapping Remote Secure Key Infrastructure [RFC8995]. 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 is required to revoke certificates before their lifetime expires.

data plane: The counterpoint to the ACP VRF in an ACP node: the forwarding of user traffic in non-autonomous nodes and/or networks and 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 details.

device: A physical system or physical node.

enrollment: The process by which a node authenticates itself to a network with an initial identity, which is often called an Initial Device IDentity (IDevID) certificate, and acquires from the network a network-specific identity, which is often called an 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 Standards 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 and/or OAM or autonomic functions. See [RFC8990].

IDevID: An Initial Device IDentity X.509 certificate installed by the vendor on new equipment. The IDevID certificate contains information that establishes the identity of the node in the context of its vendor and/or manufacturer such as device model and/or 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 (as in management or signaling): In-band management traffic and/or control plane signaling uses the same network resources such as routers and/or switches and network links that it manages and/or controls. In-band is the standard management and signaling mechanism in IP networks. Compared to out-of-band, the in-band mechanism requires no additional physical resources, but it introduces potentially circular dependencies for its correct operations. See Section 1.

Intent: The policy language of an Autonomic Network according to [RFC8993].

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 and/or manufacturer or delegated cloud service on the Internet used as part of the BRSKI protocol.

MIC: Manufacturer Installed Certificate. A synonym for 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. A node can be virtual or physical. Physical nodes are called devices.

Node-ID: The identifier of an ACP node inside that ACP. It is either the last 64 bits (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): "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." [OP-TECH]. In most cases today, OT networks are 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; however, higher-end network equipment now also has Ethernet ports dedicated only to management. An out-of-band network provides management access to the primary network independent of the configuration state of the primary network. See Section 1.

out-of-band network, virtual: 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 Section 1.

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].

registrar (ACP, ANI/BRSKI): An ACP registrar is an entity (software and/or person) that orchestrates 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 not defined in this document. See Section 6.11.7. Renewal and other maintenance (such as revocation) of ACP certificates may be performed by entities other 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 RPL. 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 a synonym of IDevID in referenced material. This term is not used in this document.

TA: Trust Anchor. A TA is an entity that is trusted for the purpose of certificate validation. TA information such as self-signed certificate(s) of the TA 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 consists of at least a device model and/or type and a serial number, often in a vendor-specific format. See sUDI and LDevID.

ULA (Global ID prefix): A Unique Local Address is an IPv6 address in the block fc00::/7, defined in "Unique Local IPv6 Unicast Addresses" [RFC4193]. ULA is the IPv6 successor of the IPv4 private address space ("Address Allocation for Private Internets" [RFC1918]). ULAs 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 is 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. 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 Provider Edge ("BGP/MPLS IP Virtual Private Networks (VPNs)" [RFC4364]) or LISP xTR ("The Locator/ID Separation Protocol (LISP)" [RFC6830]), the ACP VRF does not have any special "core facing" functionality or routing and/or mapping protocols shared across multiple VRFs. In vendor products, a VRF such as the ACP VRF may also be referred to as a 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 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 "Autonomic Networking Use Case for Distributed Detection of Service Level Agreement (SLA) Violations" [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 an Unconfigured Network

Today, bootstrapping a new node typically requires all nodes between a controlling node such as an SDN controller (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 and/or provisioning intervention. The ACP also offers additional security for any bootstrap mechanism because it can provide the encrypted discovery (via ACP GRASP) of registrars or other bootstrap servers by bootstrap proxies connecting to nodes that are to be bootstrapped. 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. Permanent Reachability Independent of the Data Plane

Today, most critical control plane protocols and OAM protocols use the data plane of the network. This leads to often undesirable dependencies between the 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 and/or control plane work as expected.

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

Data plane dependencies also affect applications in a NOC such as SDN controller applications: certain network changes are hard to implement today because the change itself may affect reachability of the devices. Examples include 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 depend on the ability to forward their packets via the data plane: sending aliveness and routing protocol signaling packets across the data plane to verify reachability, using IPv4 signaling packets for IPv4 routing and 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, and 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 they also have value on their own.

ACP2: The ACP must have a separate address space from the data plane. This separate address space provides traceability, ease of debugging, separation from data plane, and infrastructure security (filtering based on known address space).

ACP3: The ACP must use an autonomically managed address space. An autonomically managed address space provides ease of bootstrap and setup ("autonomic"), and robustness (the administrator cannot break network easily). This document uses 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.

The explanation for ACP4 is as follows: in a fully Autonomic Network (AN), all newly written ASAs could potentially communicate with each other exclusively via GRASP, and if that were the only requirement for 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-autonomous networks, it is crucial to support IPv6-layer connectivity across the ACP to support any transport-layer 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 ACP1). 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 it introduces a dependency on stable and/or 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 following 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 to 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 to 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 explicitly configured ones) are reflected in the configuration of the node.
- Non-ACP network management systems (NMS) or SDN controllers have to be explicitly configured for connection to the ACP.
- Additional candidate peer adjacencies for ACP connections across non-ACP Layer 3 clouds requires explicit configuration. See Section 8.2.

Figure 1 illustrates the ACP.

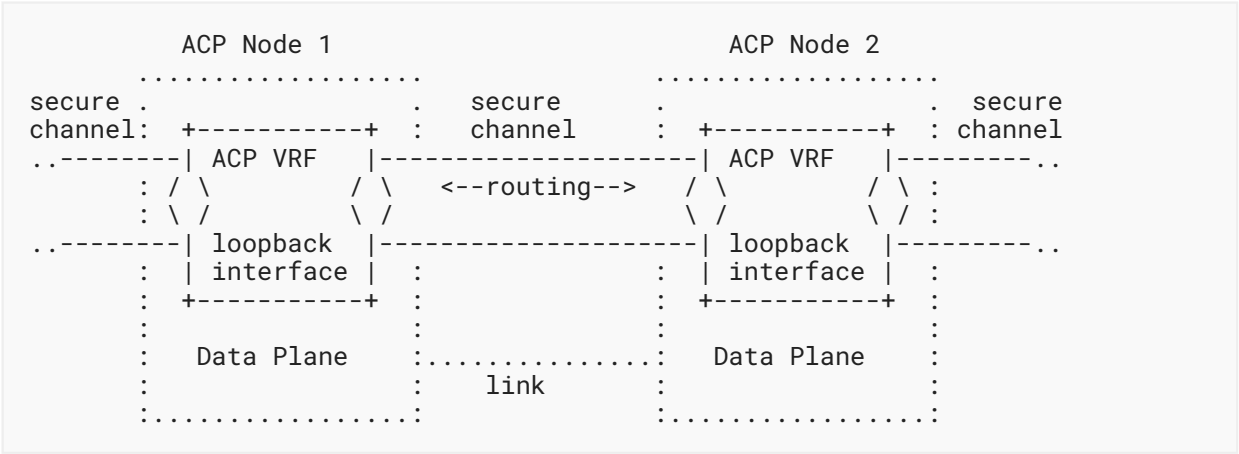

Figure 1: ACP VRF and Secure Channels

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

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

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

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

Attacks against the network via incorrect routing or addressing information for the data plane will not impact the ACP. Even impaired ACP nodes will have a significantly reduced attack surface against malicious misconfiguration because only very limited ACP or interface up/down configuration can affect the ACP, and depending on their specific designs, these types 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 the Use of Transport Layer Security (TLS)

The following requirements apply to TLS that is required or used by ACP components. Applicable ACP components include ACP certificate maintenance via EST (see Section 6.2.5), TLS connections for 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 "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)" [RFC7525] except that TLS 1.2 ("The Transport Layer Security (TLS) Protocol Version 1.2" [RFC5246]) is **REQUIRED** and that older versions of TLS **MUST NOT** be used. TLS 1.3 ("The Transport Layer Security (TLS) Protocol Version 1.3" [RFC8446]) **SHOULD** be supported. The choice for TLS 1.2 as the lowest common denominator for the ACP is based on the currently expected and 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, and it **MUST** support the NIST P-256 (secp256r1(22)) and P-384 (secp384r1(24)) curves "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier" [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 an LDevID certificate and information about one or more TAs 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 mechanism to authenticate ACP domain membership in an autonomous, peer-to-peer fashion for current and future ACP domain members.

The LDevID certificate is henceforth called the ACP certificate. The TA is the 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 that the certificate comply with Section 6.2.1, specifically that it 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 [RFC8993] use the word autonomic. This is done because those reference documents consider (only) fully Autonomic Networks and nodes, but the support of ACP does not require the support for other components of Autonomic Networks except for the reliance on GRASP and the providing of 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, [RFC8993] 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 Cryptography (ECC) public keys.

ACP nodes **MUST NOT** support certificates with RSA public keys of less than a 2048-bit modulus or curves with group order of less than 256 bits. 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 Elliptic Curve Digital Signature Algorithm (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 types 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**-level requirements 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-to-peer across the ACP require end-to-end/any-to-any authentication and 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 the 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 6.8) and for ACP GRASP (Section 6.9.2) end to end via TLS.

The ACP domain membership check requires a minimum number 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 Elliptic Curve Diffie-Hellman (ECDH) directly with the key in the ACP certificate, ACP certificates with ECC keys need to indicate that they are ECDH capable: if the X.509 v3 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 CAs.

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 the appropriate ACP registrar and/or ACP CA procedures. For example, other fields may be required for purposes other than those 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 "serialNumber" attribute ([X.520], Section 6.2.9) in the subject field distinguished name encoding. Note that this is not the certificate serial-number. See also [RFC8995], 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] because, like LLDP-signaled information, the ACP certificate information can be retrieved by neighboring nodes without further authentication and can 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 more quickly 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 Section 6.2.5 for the additional check against the id-kp-cmcRA extended key usage attribute ("Certificate Management over CMS (CMC) Updates" [RFC6402]), and see Appendix A.9.5 for possible future extensions.

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], Appendix B.1
rsub = [ <subdomain> ] ; <subdomain> as of [RFC1034], Section 3.5
acp-domain-name = <domain> ; as of [RFC1034], Section 3.5
extensions = *( "+" extension )
extension  = 1*etext  ; future standard definition.
etext      = ALPHA / DIGIT /  ; Printable US-ASCII
             "!" / "#" / "$" / "%" / "&" / "'" /
             "*" / "-" / "/" / "=" / "?" / "^" /
             "_" / "`" / "{" / "|" / "}" / "~"

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

Figure 2: ACP Node Name ABNF

Example:

Given an ACP address of fd89:b714:f3db:0:200:0:6400:0000, an ACP domain name of acp.example.com, and an rsub extension of area51.research, then this results in the following:

```
acp-node-name      = fd89b714f3db0000200000064000000
                     +area51.research@acp.example.com
acp-domain-name    = acp.example.com
routing-subdomain  = area51.research.acp.example.com
```

The acp-node-name in Figure 2 is the ABNF definition ("Augmented BNF for Syntax Specifications: ABNF" [RFC5234]) of the ACP Node Name. An ACP certificate **MUST** carry this information. It **MUST** contain an otherName field in the X.509 Subject Alternative Name extension, and the otherName **MUST** contain an AcpNodeName as described in Section 6.2.2.

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. The acp-address field is case insensitive because ABNF HEXDIG is. It is recommended to encode acp-address with lowercase 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 zero-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.

The 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. The 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 as 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 ACPs 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 picking 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 for 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.

The 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 Section 6.11.2). 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 The rsub component 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-parsable 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 were no single string representation.

2.3 If the encoding was not 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 identifiers 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 uses besides the ACP. Therefore, the information element required for the ACP should be encoded so that it minimizes the possibility of creating incompatibilities with other such uses. The attributes of the subject field, for example, are often used in non-ACP applications and therefore should not be occupied by new ACP values.

3.2 The element should not require additional ASN.1 encoding and/or decoding because libraries for accessing 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 "Extensible Messaging and Presence Protocol (XMPP): Core" [RFC6120], "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)" [RFC7585], "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name" [RFC4985], "Internationalized Email Addresses in X.509 Certificates" [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 as 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 "New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)" [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(97) }

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 10 }

  AcpNodeName ::= IA5String (SIZE (1..MAX))
   -- AcpNodeName as specified in this document carries the
   -- acp-node-name as specified in the ABNF in Section 6.2.2

END
```

Figure 3: AcpNodeName ASN.1 Module

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, Section 2.15 of "Internet Key Exchange Protocol Version 2 (IKEv2)" [RFC7296].

2. The peer's certificate passes certificate path validation as defined in [RFC5280], Section 6, against one of the TAs associated with the ACP node's ACP certificate (see Section 6.2.4). This includes verification of the validity (lifetime) of the certificates in the path.
3. If the peer's certificate indicates a CRLDP ([RFC5280], Section 4.2.1.13) or 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's 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 has no access an OCSP responder, and the security association protocol itself also has no way to communicate the CRL or OCSP check.

   Retries to learn revocation via OCSP or 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 is 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 certificate's acp-address field is used for nodes that can and should determine their ACP address through mechanisms other 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 and authorization (hence the name "routing subdomain").

In summary:

- Steps 1 through 4 constitute standard certificate validity verification and private key authentication as defined by [RFC5280] and security association protocols (such as IKEv2 [RFC7296]) when leveraging certificates.
- Except for its public key, Steps 1 through 4 do not include the verification of any preexisting form of a certificate's identity elements, such as a web server's domain name prefix, which is often encoded in the certificate common name. Step 5 is an equivalent step for the AcpNodeName.
- Step 4 constitutes standard CRL and OCSP checks refined for the case of missing connectivity and limited-functionality security association protocols.
- Steps 1 through 4 authorize the building of 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 through 5 therefore authorize the building of an ACP secure channel.

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

6.2.3.1. Realtime Clock and Time Validation

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

An ACP node without such a realtime clock **MAY** ignore those timestamp 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 NTP signaled through the ACP. It then ignores timestamp 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** be learned in an unsecured fashion, for example, via NTP ("Network Time Protocol Version 4: Protocol and Algorithms Specification" [RFC5905]) over the same link-local IPv6 addresses used for the ACP from neighboring ACP nodes. ACP nodes that do provide unsecured NTP 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.

Besides the ACP domain membership check, the ACP itself has no dependency on knowing the current time, but protocols and services using the ACP, for example, event logging, will likely need to know the current time.

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 certificates 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 the renewal of TA information via EST as described in Section 6.2.5.

The required information about a TA can consist of one or more certificates as required for dealing with CA certificate renewals as explained in Section 4.4 of "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)" [RFC4210]).

A certificate path is a chain of certificates starting at the ACP certificate (the leaf and/or end entity), followed by zero or more intermediate CA certificates, and ending with the TA information, which is typically one or two self-signed certificates of the TA. The CA that signs the ACP certificate is called the assigning CA. If there are no intermediate CAs, then the assigning CA is the TA. Certificate path validation authenticates that the TA associated with the ACP permits the ACP certificate, either directly or indirectly via one or more intermediate CA.

Note that different ACP nodes may have different intermediate CAs in their certificate path and even different TA. The set of TAs 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 the ACP domain membership check Rules 1 through 3 are performed need to support the exchange not only of the ACP nodes certificates but also the exchange of the intermediate TA.

For the ACP domain membership check, ACP nodes **MUST** support certificate path validation with zero or one intermediate CA. They **SHOULD** support two intermediate CAs and two TAs (to permit migration 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 a public CA is out of scope of this document.

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

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 usable.

ACP nodes **SHOULD** remember the GRASP O_IPv6_LOCATOR parameters of the EST server with 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 is used (see [RFC8995]), 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 on its ACP address (see Section 6.2.5.1).

The EST server **MUST** present a certificate that passes the ACP domain membership check in its TLS connection setup (Section 6.2.3, rules 1 through 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 extended key usage attribute [RFC6402], 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 a separate certificate for each of these ACP domains and respond on a different transport address (IPv6 address and/or TCP port). This is easily automated on the EST server if the CA allows 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 in the ACP via GRASP Flood Synchronization (M_FLOOD) messages. See [RFC8990], Section 2.8.11 for the definition of this message type and Figure 4 for an example.

```
[M_FLOOD, 12340815, h'fd89b714f3db00002000000064000001', 210000,
    [["SRV.est", 4, 255 ],
    [O_IPv6_LOCATOR,
         h'fd89b714f3db00002000000064000001', IPPROTO_TCP, 443]]
]
```

Figure 4: GRASP "SRV.est" Objective Example

The formal definition of the objective in CDDL (see "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures" [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 [RFC8990].
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 EST server compliant with [RFC7030] because "est" is a registered service name ("Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry" [RFC6335]) for [RFC7030]. The 'objective-value' field **MUST** be ignored if present. Backward-compatible extensions to [RFC7030] **MAY** be indicated through 'objective-value'. Certificate renewal options that are incompatible with [RFC7030] **MUST** use a different 'objective-name'. Unrecognized 'objective-value' fields (or parts thereof if it is a partially understood structure) **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 an announcement is considered expired. In the example above, the ttl is 210000 msec, that is, 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 and/or 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 that 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 have ample time to investigate and resolve any problems that cause 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 the ACP domain membership check (Section 6.2.3). The CRLDP **MAY** omit the CRL verification during authentication of the peer to permit CRL retrieval by an ACP node with a revoked ACP certificate, which can allow the (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

The certificate lifetime may be set to shorter lifetimes than customary (one year) because certificate renewal is fully automated via ACP and EST. The primary limiting factor for shorter certificate lifetimes is the 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 a discussion about an example setup achieving this. See also "Support for Short-Term, Automatically Renewed (STAR) Certificates in the Automated Certificate Management Environment (ACME)" [RFC8739].

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

See Appendix A.2 for further optimizations of certificate maintenance when BRSKI can be used [RFC8995].

6.2.5.5. Reenrollment

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 reenrolling candidate ACP node.

In this role, the node maintains the TA and certificate chain associated with its ACP certificate exclusively for the purpose of reenrollment, and it attempts (or waits) to get reenrolled with a new ACP certificate. The details depend on the mechanisms and protocols used by the ACP registrars.

Please refer to Section 6.11.7 and [RFC8995] 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 reenrolling candidate ACP node attempts to enroll like a candidate ACP node (BRSKI pledge), but instead of using the ACP node's 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 reenrollment. 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 reassign 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 reenrolling candidate ACP node **MUST** reattempt reenrollment using its IDevID certificate as defined in BRSKI during the TLS connection setup.

When the BRSKI connection is attempted with either with the old ACP certificate or the IDevID certificate, the reenrolling 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 reenrolling 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 reenroll using its old keying material and attempting to reenroll with its IDevID and requesting a voucher.

When mechanisms other than BRSKI are used for ACP certificate enrollment, the principles of the reenrolling candidate ACP node are the same. The reenrolling candidate ACP node attempts to authenticate any ACP registrar peers using reenrollment protocols and/or mechanisms via its existing certificate chain and/or 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 using enrollment mechanisms that do not leverage a mechanism to authenticate the ACP registrar (such as the voucher in BRSKI), and when the injection of certificate failures could otherwise make the ACP vulnerable to

remote attacks by returning the ACP node to a "duckling" state in which it accepts enrollment 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 reenrollment until new keying material is acquired.

When using BRSKI or other protocols and/or mechanisms that support vouchers, maintaining existing TA information allows for lighter-weight reenrollment of expired ACP certificates, 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 or 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. To determine that the ACP certificate is the culprit of a connection failure, the peer should pass the domain membership check (Section 6.2.3), and connection error diagnostics should exclude other reasons for the connection failure.

This type of failure can happen during the setup or refreshment of a secure ACP channel connection or during 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.

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

An ACP node **SHOULD** support the option to determine whether its ACP certificate is failing, and when it does, put itself into the role of a reenrolling candidate ACP node as explained in 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 (the identifier of the node inside the ACP, see Section 6.11.3 and Section 6.11.5), the interface on which neighbor was discovered (by GRASP as explained below), the link-local IPv6 address of the neighbor on that interface, and the 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.

When 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, independent 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

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 [RFC8990] for its formal definition. The ACP uses one instance of DULL GRASP for every L2 interface of the ACP node to discover candidate ACP neighbors that are link-level adjacent. 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 unconfigured. Reception of packets on such otherwise unconfigured interfaces **MUST** be limited so that at first only SLAAC ("IPv6 Stateless Address Autoconfiguration" [RFC4862]) and DULL GRASP work, and then only the following ACP secure channel setup packets work, 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 MLDv2 (see "Multicast Listener Discovery Version 2 (MLDv2) for IPv6" [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 switches ("Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches" [RFC4541]) that either do not support ACP or are not ACP enabled because those switches would stop forwarding DULL GRASP packets. Switches that do not support 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 and disable ACP at the node and interface level. See Section 8.2.2 for more details about tunnels (typical non-native interfaces). See Section 7 for extending ACP 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 an 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

Synchronization 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" Objective 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 [RFC8990]
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 in Figure 6 is the protocol used to negotiate an IPsec secure channel.

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

The 'extension' parameter allows the definition of 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 is 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 IPsec connection. GRASP therefore indicates "IKEv2" and not "IPsec". If "IPsec" was used, this could mean the use of the obsolete, older version IKE (v1) ("The Internet Key Exchange (IKE)" [RFC2409]). IKEv2 has an IANA-assigned port number 500, but in Figure 6, the candidate ACP neighbor is offering ACP secure channel negotiation via IKEv2 on port 15000 (purely to show through the example that GRASP allows the indication of a port number, and it does not have to be IANA assigned).

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 "IKEv2" and "DTLS" for 'objective-value'. 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 primary use of DULL GRASP is 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.

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 and/or ASAs.

The result of the discovery is the IPv6 link-local address of the neighbor as well as its supported secure channel protocols (and the 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 described DULL GRASP objective 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 multicast 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, the initial discovery of candidate ACP peers cannot include any unprotected negotiation. To avoid reinventing and validating security association mechanisms, the next step after discovering the address of a candidate neighbor is to establish a security association with that neighbor using a well-known security association method.

It seems clear from the use cases that not all types of ACP nodes can or need to connect directly to each other, nor are they able to support or prefer all possible mechanisms. For example, IoT devices that are codespace limited 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 an interesting potential secure channel protocol. Note also that the security model allows and requires any-to-any authentication and authorization between all ACP nodes because there is not only hop-by-hop but also end-to-end authentication for secure channels.

To support extensible selection of the secure channel protocol without a single common mandatory-to-implement (MTI) protocol, an ACP node **MUST** try all the ACP secure channel protocols it supports and that are also announced by the candidate ACP neighbor 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, continuing to attempt the setup of 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 Decider's 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 connection via IKEv2 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 more preferred protocol options (in its view, 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 to 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 afterward.

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] Node 1 and Node 2 have authenticated each other's certificate on connection C1 as valid ACP peers.

[8:C1] Node 1's certificate has a lower ACP Node-ID than Node 2's, 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 2.

[10:C2] Node 1 and Node 2 have authenticated each other's certificate on connection C2 (like [7:C1]).

[11:C2] Node 1's certificate has a lower ACP Node-ID than Node 2's, 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.

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 may the supported ACP secure channel protocol options 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-two 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 describes how peers that are adjacent on an IPv6 subnet are discovered automatically. Section 8.2 describes how to configure peers that are not adjacent on an IPv6 subnet.

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 to 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 multiparty 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 and/or 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 Layer 3 (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 that leverage such underlying L2 security to auto-discover and associate ACP peers 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 specifies with ACP connect (Section 8.1) only 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 neighbor's 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 AES-256. 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 relies 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 revalidated 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, any protocol options that are unnecessary for the support of devices that are expected to be able to support the ACP **SHOULD** be eliminated in order to minimize implementation complexity. For example, definitions for security protocols often include old and/or inferior security options required only to interoperate with existing devices that cannot update to the currently preferred security options. Such old and/or inferior security options do not need to be supported when a security association protocol is first specified for the ACP, thus 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 ("Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)" [RFC8221]) and IKEv2 ("Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)" [RFC8247]). These options result in stringent security properties and can exclude deprecated and legacy algorithms because there is no need for interoperability with legacy equipment for ACP secure channels. Any such backward compatibility would lead only to an 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 and/or 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. RFC 8221 (IPsec/ESP)

ACP IPsec implementations **MUST** comply with [RFC8221] and any specifications that update it. The requirements from above and this section amend and supersede 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 and/or 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 with a potentially big drop in performance could also render the ACP inoperable. Therefore, there is a 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.

6.8.3.1.2. RFC 8247 (IKEv2)

[RFC8247] provides a baseline recommendation for mandatory-to-implement ciphers, integrity checks, pseudorandom functions, and Diffie-Hellman mechanisms. Those recommendations, and the recommendations of subsequent documents, apply as well to the ACP. Because IKEv2 for ACP secure channels is sufficient to be implemented in control plane software rather than in Application-Specific Integrated Circuit (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 document 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, the "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 it does not indicate any of this ACP node's 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 it to obtain the diagnostic information about the neighbor's certificate.

In IKEv2, ACP nodes are identified by their ACP addresses. 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 (modular exponentiation (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 the IKEv2 initiator and responder support IPsec over GRE, it will be preferred over native IPsec because of how IKEv2 negotiates transport mode (as used by this IPsec over GRE profile) versus tunnel mode as used by native IPsec (see Section 1.3.1 of [RFC7296]). The ACP IPv6 traffic has to be carried across GRE according to "IPv6 Support for Generic Routing Encapsulation (GRE)" [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 ("Datagram Transport Layer Security 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 1.2 connection in the presence of a peer that only supports DTLS 1.2.

All ACP nodes supporting DTLS as a secure channel protocol **MUST** adhere to the DTLS implementation recommendations and security considerations of BCP 195 [RFC7525] except with respect to the DTLS version. ACP nodes supporting DTLS **MUST** 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 recommendations in BCP 195 [RFC7525] because the expectation is that DTLS would use software-only implementations where the ability to reuse widely adopted implementations is more important than the ability to minimize the complexity of a hardware-accelerated implementation, which is known to be important for IPsec.

DTLS 1.3 [TLS-DTLS13] is "backward compatible" with DTLS 1.2 (see Section 1 of [TLS-DTLS13]). A DTLS implementation supporting both DTLS 1.2 and DTLS 1.3 does comply with the above requirements of negotiating to DTLS 1.2 in the presence of a DTLS 1.2 only peer, but using DTLS 1.3 when booth peers support it.

Version 1.2 is the MTI version of DTLS in this specification because of the following:

- There is more experience with DTLS 1.2 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 with DTLS 1.3, such as a different record layer requiring time to gain implementation and deployment experience especially on lower-end devices with limited code space.
- The existing BCP [RFC7525] for DTLS 1.2 may take an equally longer time to be updated with experience from a newer DTLS version.
- There are no significant benefits of DTLS 1.3 over DTLS 1.2 that are use-case relevant in the context of the ACP options for DTLS. For example, signaling performance improvements for session setup in DTLS 1.3 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 1.3, have stricter security requirements, and the use of the latest standard protocol version is in general recommended for IETF security standards. Therefore, ACP implementations are advised to support all the newer versions of DTLS that can still negotiate down to DTLS 1.2.

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 rekeying 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 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 both IPsec and DTLS and therefore supports both the baseline and constrained profiles.

Explanation: not all types of ACP nodes are able to or need to connect directly to each other, nor are they able to support or prefer all possible secure channel mechanisms. For example, IoT devices with limited code space may only support DTLS because that code already exists 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 they 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 types of nodes as long as there are also appropriate gateway ACP nodes that sufficiently support 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 are the norm already.

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

ACP nodes need to specify the set of secure ACP mechanisms they support in documentation and should declare which profile they support according to the 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 RPL (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 [RFC8990], 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 it continues to exist even if all of its neighbors cease operation.

In this case, ASAs using GRASP running on the same node 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 they **MUST** be able to understand that there is no ACP and that therefore the ACP instance of GRASP cannot operate.

How the ACP acts as the security and transport substrate for GRASP is shown in Figure 8.

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 they are sent by the ACP to 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 (7017). Effectively, the transport stack is expected to be TLS for connections to and from the ACP address (e.g., global-scope address(es)) and TCP for connections to and from the link-local addresses on the ACP virtual interfaces. The latter ones are only used for the flooding of GRASP messages.

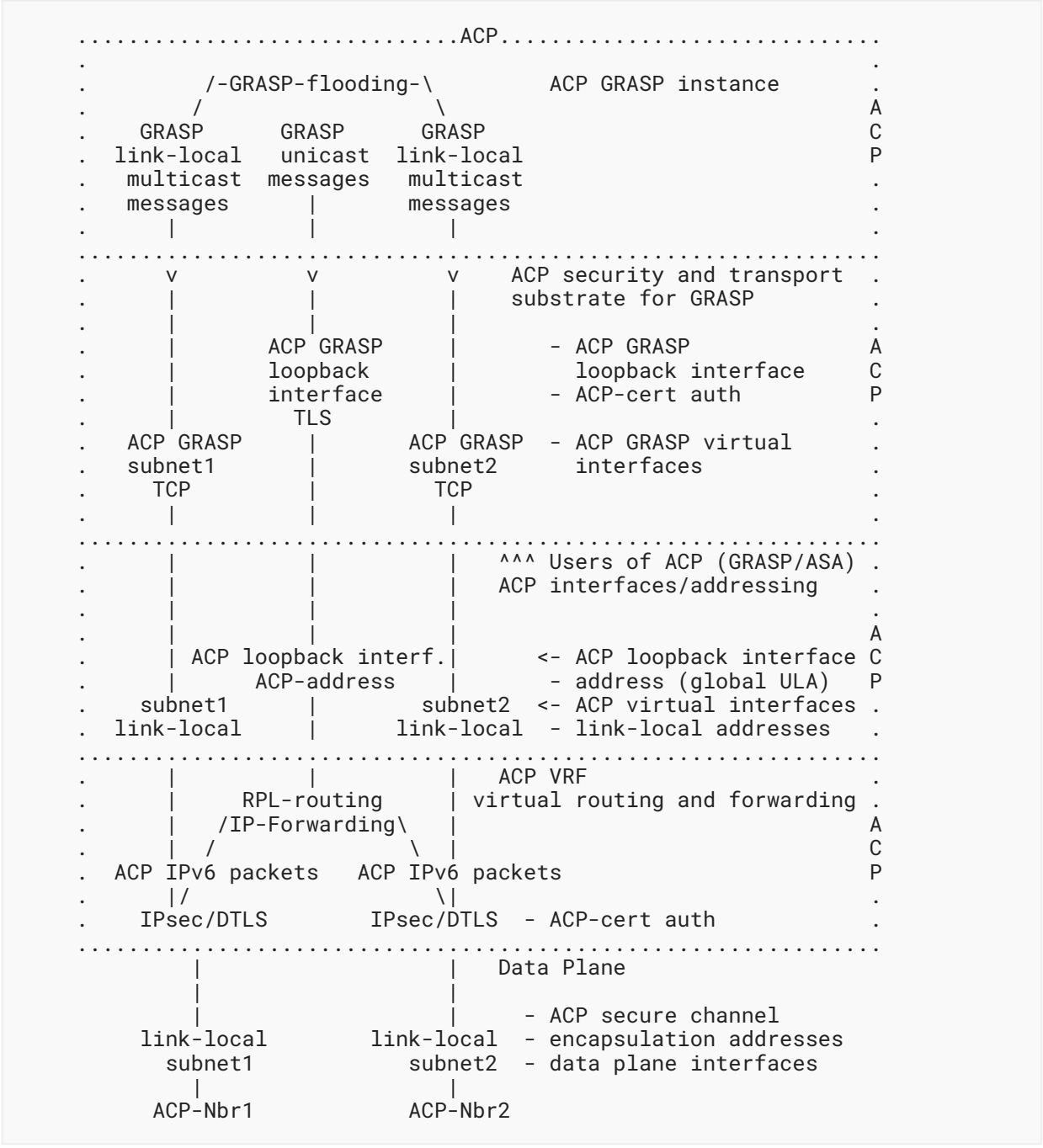

Figure 8: ACP as Security and Transport Substrate for GRASP

6.9.2.1. Discussion

TCP encapsulation for GRASP M_DISCOVERY and M_FLOOD link-local messages is used because these messages are flooded across potentially many hops to all ACP nodes, and a single link with even temporary packet-loss issues (e.g., a Wi-Fi or 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 offered 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 and/or expiry).

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

Eavesdropping and/or 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 that 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 the compromised node to proxy the messages 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 and 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.
- The security model for ACP GRASP is no different than 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 a classical network system, a dedicated VRF is one logical implementation option for the ACP. If allowed by the system's software architecture, separation options that minimize shared components, such as a logical container or virtual machine instance, are preferred. The context

for the ACP needs to be established automatically during the 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 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, a 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 "Using Only Link-Local Addressing inside an IPv6 Network" [RFC7404].
- Use of ULA: for loopback interfaces of ACP nodes, we use ULA with the L bit set to 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 in [RFC4193], Section 3.2.2.
- No external connectivity: the addresses do not provide access to the Internet. If a node requires further connectivity, it should use another, traditionally managed addressing scheme in parallel.
- Addresses in the ACP are permanent and do not support temporary addresses as defined in "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6" [RFC8981].
- Addresses in the ACP are not considered sensitive on privacy grounds because ACP nodes are not expected to be end-user hosts, and therefore ACP addresses do not represent end users

or groups of end users. All ACP nodes are in one (potentially federated) administrative domain. For ACP traffic, the nodes are assumed to be either candidate hosts or transit nodes. There are no transit nodes with fewer privileges to know the identity of other hosts in the ACP. Therefore, ACP addresses do not need to be pseudorandom as discussed in "Security and Privacy Considerations for IPv6 Address Generation Mechanisms" [RFC7721]. Because they are not propagated to untrusted (non-ACP) nodes and stay within a domain (of trust), we also do not consider them 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, etc.) 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 ULA addressing base scheme for ACP nodes has the following format:

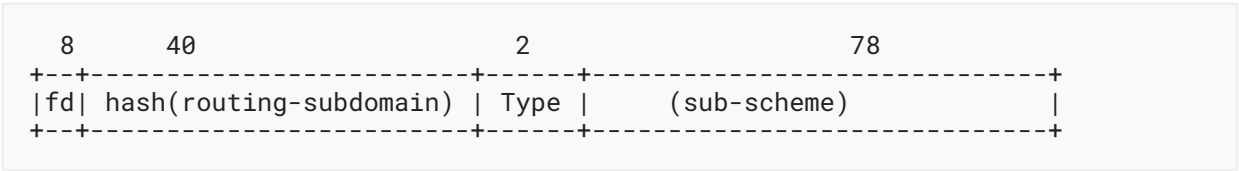

Figure 9: ACP Addressing Base Scheme

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

fd: Identifies a locally defined ULA address.

hash(routing-subdomain): The 40-bit ULA Global ID (a term from [RFC4193]) for ACP addresses carried in the acp-node-name in the ACP certificates are the first 40 bits of the SHA-256 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-bit ULA Global ID is 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 preexisting 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) Attributes Request message sent by the pledge.

Establishing connectivity between different ACPs (different acp-domain-names) 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 needs to be selected so that 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 do not connect but where the option to do so should be kept open. When taking this possibility into account, it is always easy to select rsub so that no collisions happen.

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

(sub-scheme): 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 for why the following addressing sub-schemes are used and why multiple are necessary.

The following summarizes the addressing sub-schemes:

Type	Name	F-bit	Z	V-bits	Prefix
0	ACP-Zone	N/A	0	1 bit	/127
0	ACP-Manual	N/A	1	N/A	/64
1	ACP-Vlong-8	0	N/A	8 bits	/120
1	ACP-Vlong-16	1	N/A	16 bits	/112
2	Reserved / For future definition/allocation				
3	Reserved / For future definition/allocation				

Table 1: Addressing Sub-Schemes

The F-bit (format bit, Section 6.11.5) and Z (Section 6.11.4) are two encoding fields that are explained in the sections covering the sub-schemes that use them. V-bits is the number of bits of addresses allocated to the ACP node. Prefix is the prefix that the ACP node is announcing into RPL.

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

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

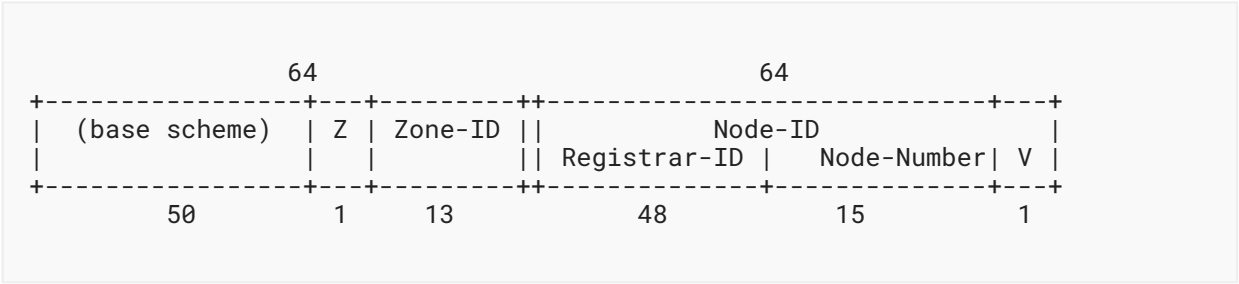

Figure 10: ACP Zone Addressing Sub-Scheme

The fields are defined as follows:

Type: **MUST** be 0.

Z: **MUST** be 0.

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 bits): A number unique inside the domain identifying the ACP registrar that 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: A 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 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. Therefore, no two nodes in the same ACP and the same hash(routing-subdomain) can have the same Node-ID with the Zone Addressing Sub-Scheme, for example, by differing only in their Zone-ID.

The Virtualization 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 preexisting 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 (peer-to-peer) virtual interface with the V=0 address as its router to 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 avoids 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 or incremental adoption of the ACP as described in Section 9.4.

Note: Zones and Zone-ID as defined here are not related to zones or zone_id defined in "IPv6 Scoped Address Architecture" [RFC4007]. ACP zone addresses are not scoped (i.e., reachable only from within a zone as defined by [RFC4007]) but are reachable across the whole ACP. A zone_id is a zone index that has only local significance on a node [RFC4007], 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 0 and the Z bit is 1.

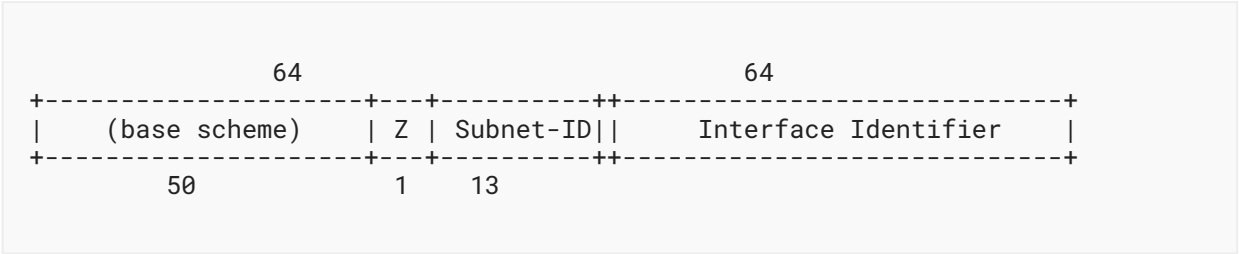

Figure 11: ACP Manual Addressing Sub-Scheme

The fields are defined as follows:

Type: **MUST** be 0.

Z: **MUST** be 1.

Subnet-ID: Configured subnet identifier.

Interface Identifier: Interface identifier according to [RFC4291].

This sub-scheme is meant for the "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 with preexisting, 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 between the Zone Addressing Sub-Scheme and the Manual Addressing Sub-Scheme without requiring one more bit in the base scheme and therefore allowing for the Vlong Addressing Sub-Scheme (described in Section 6.11.5) to have one more bit available.

The Manual Addressing Sub-Scheme addresses **SHOULD NOT** be used in ACP certificates. Any node capable of building ACP secure channels and is 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. A node that cannot or is not 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. Its ACP certificate **MUST** omit the acp-address field to indicate that its 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-bits multiple 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 addressing sub-scheme is used when the Type field of the base scheme is 1.

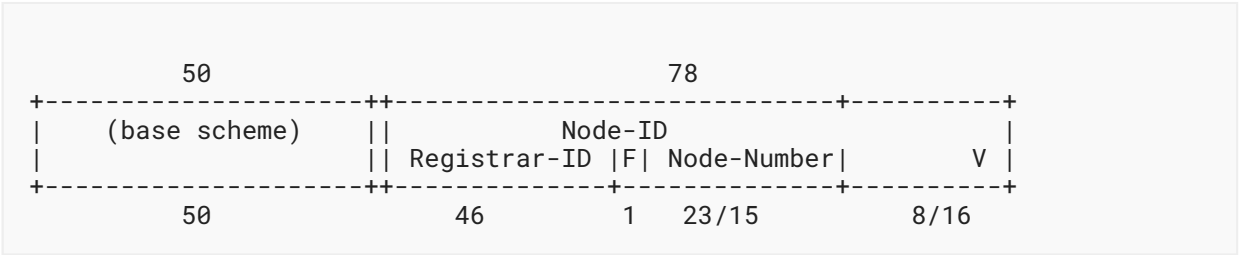

Figure 12: ACP Vlong Addressing Sub-Scheme

This addressing sub-scheme foregoes the Zone-ID field (Section 6.11.3) to allow for larger, flatter routed networks (e.g., as in IoT) with 8,421,376 Node-Numbers ($2^{23} + 2^{15}$). It also allows for up to 2^{16} (i.e., 65,536) 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 Addressing Sub-Scheme (Section 6.11.3) 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.

Node-Number: 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 (less than 256) of clients (ASA and/or 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 a sufficiently flexible setup of ACPs for a variety of situations. 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-IDs. 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 for enrolling candidate ACP nodes with ACP certificates and associated trust anchor(s). They are also responsible for including an acp-node-name field in the ACP certificate. This field carries the ACP domain name and the ACP node's 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 CA through any appropriate mechanism (out of scope in this document, but this mechanism is required to be BRSKI for ANI registrars). Only nodes that are trusted to be compliant with the registrar requirements described in this section can be given the necessary credentials to perform this RA function, such as the credential for the ACP registrar to connect to the CA as a registrar.

6.11.7.1. Use of BRSKI or Other Mechanisms or Protocols

Any protocols or mechanisms may be used by ACP registrars as long as the resulting ACP certificate and TA certificate(s) can be used by other domain members to perform the ACP domain membership check described in Section 6.2.3, and the acp-node-name meets the ACP addressing requirements described in the next three sections.

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 [RFC8995]. 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 or mechanisms that 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, called the Registrar-ID, that are unique across the ACP domain. Allocation of Registrar-ID(s) to an ACP registrar can happen through OAM mechanisms in conjunction with some database and/or allocation orchestration.

ACP registrars running on physical devices with known globally unique EUI-48 MAC address(es) (EUI stands for "Extended Unique Identifier") can use the lower 46 bits of those address(es) as unique Registrar-IDs without requiring any external signaling and/or configuration (the upper two bits, V and U, are not uniquely assigned but are 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 or Network Interface Controller (NIC) 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 an 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 EST CSR Attributes (see [RFC8995], Section 5.9.2, "EST CSR Attributes").

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 addressing sub-scheme indicates the prefix length: /127 for the Zone Addressing Sub-Scheme, /120 or /112 for the Vlong Addressing 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 Addressing Sub-Scheme 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 Addressing 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 often indicates the node's vendor and device type, and it can be used to drive a policy for selecting an appropriate addressing sub-scheme for the (class of) node(s).

ACP registrars **SHOULD** default to allocating Zone Addressing Sub-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 Addressing Sub-Scheme for ACP nodes based on the "serialNumber" attribute [X.520] 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 by the complete "serialNumber". The PID, for example, could identify nodes that allow for specialized ASA requiring multiple addresses or for non-autonomic virtual machines (VMs) for services, and those nodes could receive Vlong Addressing Sub-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 it increases the next Node-Number during successful enrollment of an ACP node. In this simple allocation scheme, the ACP registrar would not recycle ACP address prefixes from ACP nodes that are no longer used.

If allocated addresses cannot be remembered by registrars, then it is necessary either to use a new value for the Register-ID field in the ACP addresses or to determine allocated ACP addresses by determining the addresses of reachable ACP nodes, which is not necessarily the set of all ACP nodes. Untracked ACP addresses can be reclaimed by revoking or not renewing their certificates and instead handing out new certificates 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 and/or 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 expires or fails. 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: needed interactions, required parameters, certificate renewal and limitations, use of sub-CAs on registrars, and centralized policy control.

6.12. Routing in the ACP

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

The establishment of the routing plane and its parameters are automatic and strictly within the confines of the ACP. 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 [ROLL-APPLICABILITY].

6.12.1.1. Overview

RPL Packet Information (RPI), defined in [RFC6550], Section 11.2, defines the data packet artifacts required or beneficial in the forwarding of packets routed by RPL. This profile does not use RPI for better compatibility with accelerated hardware forwarding planes, which most often do 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 and/or removing them.

6.12.1.1.1. Single Instance

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

This choice is a compromise. Consider a network that has multiple NOCs in different locations. Only one NOC will become the DODAG root. Traffic to and from other NOCs has to be sent through the DODAG (shortest path tree) rooted in the primary NOC. Depending on topology, this can be an annoyance from a point of view of latency or 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 nodes reachable through the same interface as the DODAG root. This RPL profile can therefore scale to a much larger number of ACP nodes in the same amount of computation 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 RPI (see Section 6.12.1.13) is present, it is also used to trigger reconvergence when misrouted, for example, looping packets, which 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 quickly 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 signaling packets should be exceedingly rare.

In addition, there are a variety of mechanisms possible in RPL to further avoid temporary loops that are **RECOMMENDED** to be used for the ACP RPL profile: DODAG Information Objects (DIOs) **SHOULD** be sent two or three 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 three if possible, and send Destination Advertisement Objects (DAOs) 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 Network's (LLN's) Expected Transmission Count (ETX) feature, which 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

There is a single RPL instance. The default RPLInstanceID is 0.

6.12.1.3. Storing vs. Non-Storing Mode

The RPL Mode of Operation (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

The DAO policy is proactive, aggressive DAO state maintenance:

- Use the 'K' flag in unsolicited DAO to indicate 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 Metrics

Use Hop Count according to "Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks" [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 Objective Function Zero (OF0) ("Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL)" [RFC6552]). Metric containers are not used.

rank_factor: Derived from link speed: if less than or equal to 100 Mbps, LOW_SPEED_FACTOR(5), else HIGH_SPEED_FACTOR(1).

This is a simple rank differentiation between typical "low speed" or IoT links that commonly max out at 100 Mbps and typical infrastructure links with speeds of 1 Gbps or higher. Given how the path selection for the ACP focuses 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 sends a 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 a new DIO that advertises 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

Multicast is not used yet, but it is possible because of the selected mode of operations.

6.12.1.9. Security

RPL security [RFC6550] is not used, and ACP security is substituted.

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 the Zone Addressing Sub-Scheme and /112 or /120 for the Vlong Addressing Sub-Scheme. See Section 6.11 for more details.

Every ACP node **MUST** install a black hole route (also known as a null route) if there are unused parts of the ACP address space assigned to the ACP node via its AcpNodeName. This is superseded by longer prefixes assigned to interfaces for the address space actually used by the node. For

example, when the node has an ACP-Vlong-8 address space, it installs a /120 black hole route. If it then only uses the ACP address (first address from the space), for example, 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], Section 3.2.6 --to become root): The preference is indicated in the DODAGPreference field of DIO message.

Explicitly 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) ("An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)" [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 "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams" [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 established between adjacent neighbors by default, 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 for ACP implementations defined in this document because the performance requirements depend on the intended use case. It is expected that a fully autonomic node with a wide range of ASA can require high forwarding plane performance in the ACP, for example, for telemetry. Implementations of ACP that solely support traditional or 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 the design 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 ACP secure channels discovered via GRASP 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 impacting the operations of the IPv6 (link-local) interface/address used for ACP channels when configuring the data plane, 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 and/or 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 (nonideal) implementation choices may introduce additional, undesired dependencies against the data plane, for example, shared code and configuration of the secure channel protocols (IPsec and/or 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 MTUs 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 octets) 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 subset 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 retry to build these secure channels from its side unless the Decider appears 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

Conceptually, the ACP VRF has two types 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 assigned to the node. N = (128 - number of Vbits of the ACP address). 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 do not require additional global-scope IPv6 addresses, they are instead intended for ASA or non-autonomous functions. Components of the ACP that are not fully autonomic, such as ACP connect interfaces (see Figure 14), may also introduce additional global-scope IPv6 addresses on other types of interfaces to the ACP.

The use of loopback interfaces for global-scope addresses is common operational configuration practice on routers, for example, in Internal BGP (IBGP) connections since BGP4 (see "A Border Gateway Protocol 4 (BGP-4)" [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 that behaves according to Section 4 of "Default Address Selection for Internet Protocol Version 6 (IPv6)" [RFC6724], 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 term "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 and/or 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 list reviews the reasons for the choice of loopback addresses for ACP addresses, which 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 Section 2.1 of "IP Version 6 Addressing Architecture" [RFC4291].
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 connect 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 bring down the interface and make 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 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 at a time for a connection with the most widely deployed transport protocols, TCP and UDP. While "TCP Extensions for Multipath Operation with Multiple Addresses" [RFC6824]/[RFC8684] solves this problem, it is not widely adopted by implementations of router OAM services.
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 on 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 which 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 increases the size of the 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 sub-schemes do not consider ACP addresses for subnets connecting ACP nodes.

Note that "Segment Routing Architecture" [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, either standardized or not).

Note that all the considerations described here assume point-to-point secure channel associations. Mapping multiparty secure channel associations, such as "The Group Domain of Interpretation" [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 to 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 to an ACP multi-access virtual interface are replicated to every ACP secure channel mapped to the ACP multi-access virtual interface. IPv6 unicast packets sent to 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 that a peer is dead for a secure channel mapped to 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 [RFC4862] to assign their IPv6 link-local address on the ACP virtual interface and ND ("Neighbor Discovery for IP version 6 (IPv6)" [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 "Optimistic Duplicate Address Detection (DAD) for IPv6" [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, locally 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. 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 to 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 reflect the presence of the underlying multi-access subnet to 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 interfaces on the efficiency of flooding via link-local multicast 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 packets. 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 emulated a multi-access virtual interface, then this would not happen because GRASP's flooding procedure does not replicate packets back to the interface from which they were received.

Note that link-local GRASP multicast messages are not sent directly as IPv6 link-local multicast UDP messages to ACP virtual interfaces, but instead to 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 messages and therefore have 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. As a result, GRASP message flooding would fail. Instead, each ACP secure channel across such an interface **MUST** be represented as an 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)

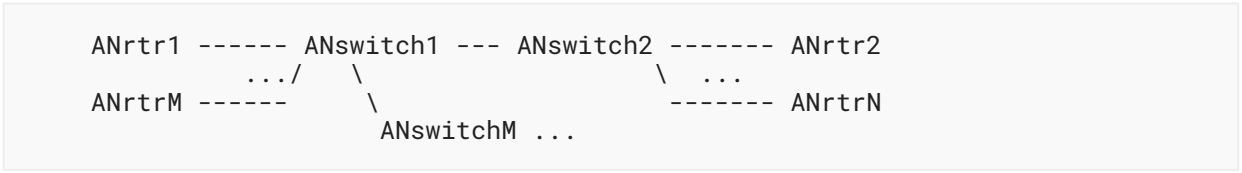

Figure 13: Topology with L2 ACP Switches

Consider a large L2 LAN with routers ANrtr1 through 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 a multilevel L2-switched topology.

If the discovery protocol used for the ACP operates 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) need to scale to the number of direct ACP neighbors. An ACP router with just four 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 that can then also be leveraged in management operations or by ASAs.

In the example above, consider that 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, and ANrtrN have a full mesh of ACP connections 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 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 the 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 forwarded 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 the logic described above 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.

As a 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 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 is 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 and 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 (STP). Without further L2 enhancements, the ACP would run only across the active STP topology, and the ACP would be interrupted and reconverge 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 reconvergence can be quite slow. The above described simple implementation options are not sufficient to achieve this.

8. Support for Non-ACP Components (Normative)

8.1. ACP Connect

8.1.1. Non-ACP Controller and/or Network Management System (NMS)

The ACP can be used by management systems, such as controllers or 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 access only to trusted, autonomic systems by default. Therefore, a traditional, non-ACP NMS 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.

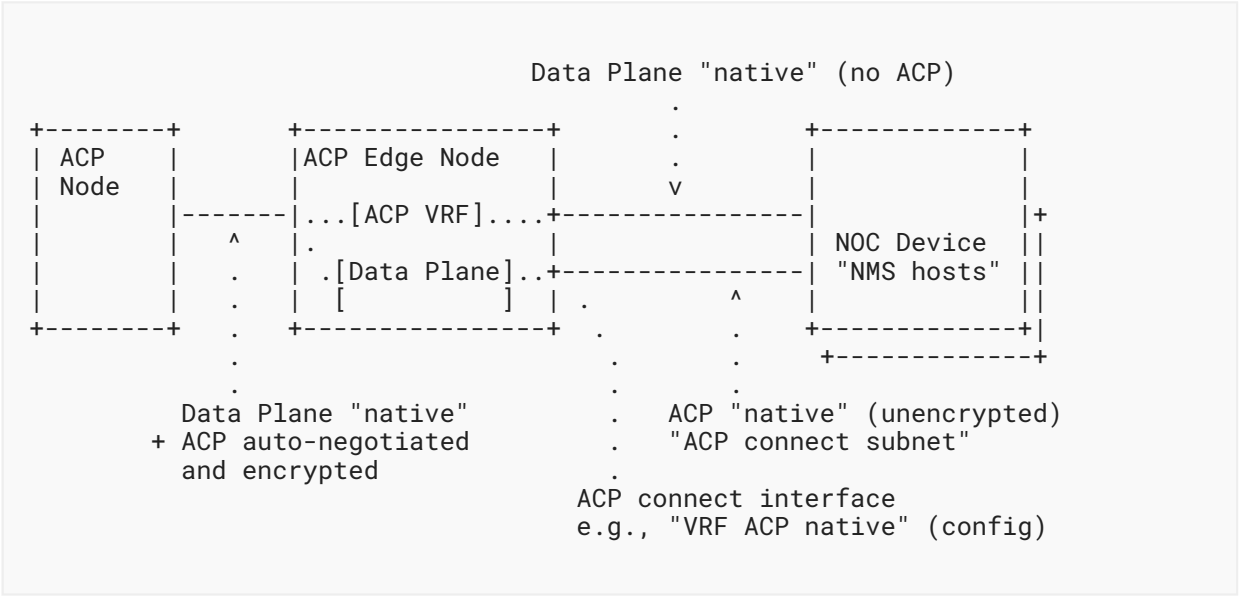

Figure 14: ACP Connect

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 the NOC systems connected to it need to be physically controlled and/or secured. For this reason, the mechanisms described here explicitly do 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.

Physically controlled and/or secured means that attackers cannot gain 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 colocated in an access-controlled room, such as a NOC, to which attackers cannot gain physical access.

An ACP connect interface provides exclusive 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 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 host **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 host's ACP connect interface uses another prefix, or if the ACP uses multiple ULA prefixes, then the NMS host requires (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 to the ACP if the packet has an IPv6 source address from that interface (this is sometimes called Reverse Path Forwarding (RPF) filtering). This filtering rule **MAY** be changed through administrative measures. The more any such administrative action enables 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 and/or 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 reenrollment. 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 the ACP edge node and NOC devices are separate physical devices and that 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 can be used not only 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 colocate an ACP edge node and an NMS host by making one a virtual machine or container inside the other; 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 host's 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 to the data plane. Like in the physical setup for NMS hosts, this can be realized via two internal virtual subnets: one that connects to the ACP (which could be a container or virtual machine by itself), and one (or more) connecting to the data plane.

This "internal" use of the 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. Autoconfiguration

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 SLAAC [RFC4862] on the ACP connect subnet and route autoconfiguration according to "Default Router Preferences and More-Specific Routes" [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 to 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 the connecting of type C hosts to the ACP via an ACP connect subnet on one interface and another network (Data Plane and/or NMS network) on the same or another interface of the type C host, relying on routers other 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 and B hosts ignore the RIOs and will consider the ACP node to be their default router for all destinations. This is sufficient when the type A or type B host only needs to connect to the ACP but not to other networks. Attaching a type A or type B host to both the ACP and other networks requires explicit ACP prefix route configuration on either the host or the combined ACP and 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 (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 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 and/or provision the address prefixes for such ACP connect interfaces.

8.1.4. Combined ACP and Data Plane Interface (VRF Select)

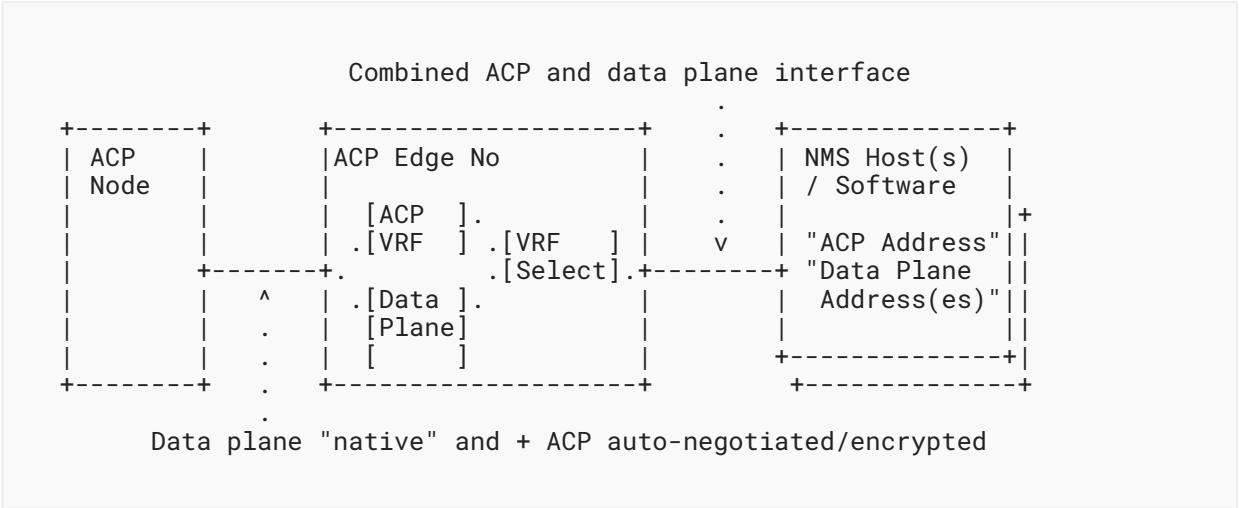

Figure 15: VRF Select

Using two physical and/or virtual subnets (and therefore interfaces) to 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 type A or type B hosts [RFC4191] (see Section 8.1.3).

To provide a single subnet to both the ACP and Data plane, the ACP edge node needs to demultiplex 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, and 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 (Section 5 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 prefix announcements [RFC4191] that are to be routed across the ACP connect interface, the source address selection of Rule 5 (use address of outgoing interface) (Section 5 of [RFC6724]) 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 in Section 8.1.3: for the ACP, the ACP prefix is announced together with the prefix option [RFC4191] routed across the ACP, and the lifetime is set to zero 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.

As a result, source address selection Rule 5.5 (Section 5 of [RFC6724]) may result in the same correct source address selection behavior of NMS hosts without further configuration as the separate ACP connect and data plane interfaces on the host. As described in the text for Rule 5.5 (Section 5 of [RFC6724]), 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 "First-Hop Router Selection by Hosts in a Multi-Prefix Network" [RFC8028] should (instead of may) implement Rule 5.5 (Section 5 of [RFC6724]), 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 architecturally 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 are that those devices connected via ACP connect are equivalently (if not better) secured against attacks than ACP nodes that do not use ACP connect, and they 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 and/or 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 the combined ACP and data plane interface, care has to be taken that only GRASP messages received from software or NMS hosts and intended for the ACP GRASP domain 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). Currently 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 L2 devices are between ACP nodes, the ACP will work across them 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 L3 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. The syntax shown is non-normative (as there are no standards for configuration) but only meant to illustrate the parameters and which ones can be optional.

```
connection = method "," local-addr "," remote-addr [ "," pmtu ]
method =    "any"
          / ( "IKEv2" [ ":" port ] )
          / (  "DTLS"  [ ":"  port ] )
port = 1*DIGIT
local-addr  = [ address  [ ":" vrf  ] ]
remote-addr =   address
address =  "any"
          / IPv4address / IPv6address  ; From [RFC5954]
vrf = system-dependent ; Name of VRF for local-address
```

Figure 16: Parameters for Remote ACP Neighbors

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

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

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

- The VRF across which the connection is built (and in which local-addr exists) can 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 the 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, for example, a hub site for remote "spoke" sites reachable over the Internet.
- The pmtu parameter should be configurable to overcome issues or 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 IP-in-IP, GRE, or other form of preexisting 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. As a 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. In such an attack, the MITM filters packets for all but the most easily attacked ACP secure channel option to force use of that 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 and 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 deploying 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 and/or requirements for the internal operational model that are beneficial or necessary to achieve the desired use-case benefits of the ACP (see Section 3).

- Section 9.1 describes the recommended capabilities of operator diagnostics of ACP nodes.
- Section 9.2 describes at a 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 removing 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, such as 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 understand how to interpret the data, which 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 into 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 remotely 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 how to capture (log) diagnostics locally for later retrieval. Ideally, these captures are also nonvolatile so that they can survive extended power-off conditions, for example, when a device that fails to be brought up zero-touch is sent for diagnostics at a more appropriate location.

The simplest form of diagnostics for answering questions such as the above is to represent the relevant information sequentially in dependency order, so that the first unexpected and/or nonoperational item is the most likely root cause, or just log and/or highlight that item. For example:

Question: Is the ACP operational to accept neighbor connections?

- Check if the necessary configurations 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 battery failure of the clock chip? Certificate checks depend on reasonable notion of time.
- Does the node have keying material, such as domain certificate, TA certificates, etc.?
- If there is no keying material and the 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? The validity start time is in the past, and the expiration time is 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 of the above looks good, the ACP should be running "fine" locally, 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? (Are there 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:
  - Did the neighbor close the connection on us, or did we close the connection on it because the domain certificate membership failed?
  - If the neighbor closed the connection on us, provide any error diagnostics from the secure channel protocol.
  - If we failed the attempt, display our local reason:
    - There was no common secure channel protocol supported by the two neighbors (this could not happen on nodes supporting this specification because it mandates common support for IPsec).
    - Did the ACP certificate membership check (Section 6.2.3) fail?
      - The neighbor's certificate is not signed directly or indirectly by one of the node's 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 acp-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 it is not yet valid.
  - Are there any other connection issues, 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 RPL 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? (If BRSKI is supported.)
- Is the BRSKI proxy operating normally on all interfaces where ACP is operating?

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

The items so far illustrate how the ANI operations can be diagnosed with passive observation of the operational state of its components including historic, cached, and/or counted events. This is not necessarily 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 pledge's IDevID certificate. Only the selected BRSKI registrar can do this, but it may be difficult to disseminate information from those BRSKI registrars about undesired pledges to locations and/or nodes where information about those pledges is desired.
2. LLDP disseminates information about nodes, such as node model, type, and/or software and interface name and/or number of the connection, to their immediate neighbors. This information is often helpful or even necessary in network diagnostics. It can equally be considered 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 also including the TA certificate in the secure channel signaling. This should be possible to do without modifying the security association protocols used by the ACP. For example, while [RFC7296] does not mention this, it also does not prohibit it.

One common use case where diagnostics through the signaled TA of a candidate peer are very helpful is the multi-tenant environment, such as an office building, 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 tenant's 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 treat such interfaces accordingly. The information in the TA of the peer can then ease troubleshooting of such issues.

Another use case is the intended or accidental reactivation of equipment, such as redundant gear taken from storage, whose TA certificate has long expired.

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

A fourth use case is when multiple registrars are set up for the same ACP but are not correctly set up with the same TA. For example, when registrars support also being CAs themselves but are misconfigured to become TAs instead of intermediate CAs.

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 provide ACP address prefixes to candidate ACP nodes by enrolling them with an ACP certificate into an ACP domain via any appropriate mechanism and/or 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 registrars 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 minimal, preexisting (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 and/or candidate ACP nodes could want to use, for example, BRSKI over CoAP (Constrained Application Protocol) or the proxying of NETCONF.

To reach a TA that has no ACP connectivity, the ACP registrar uses the data plane. The 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 an 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 the ACP registrar and the candidate ACP node to discover the other side, and then connect and enroll (configure) the candidate ACP node with an ACP certificate. For example, NETCONF Zero Touch ("Secure Zero Touch Provisioning (SZTP)" [RFC8572]) is a 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 a candidate ACP node, such as a BRSKI pledge, has secure bootstrap, it will not trust being configured and/or enrolled across the network unless it is presented with a voucher (see "A Voucher Artifact for Bootstrapping Protocols" [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 Zero Touch 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 the ACP nodes and via the data plane to the TA and other sources of CRL information.

9.2.2. Registrar Parameters

The interactions of an ACP registrar outlined in Section 6.11.7 and Section 9.2.1 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 usable Node-IDs for the Zone Addressing Sub-Scheme (Zone-ID 0) and for the Vlong Addressing Sub-Scheme (/112 and /120). 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 on whether the candidate ACP nodes should receive a domain certificate or not, for example, based on the device's IDevID certificate as in BRSKI. The ACP registrar may whitelist or blacklist based on a device's "serialNumber" attribute [X.520] in the subject field distinguished name encoding of its IDevID certificate.

- Policies on what type of address prefix to assign to a candidate ACP device, likely based on the same information.
- For BRSKI or other mechanisms using vouchers: parameters to determine how to retrieve vouchers for specific types 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 and/or 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 node's current ACP certificate and maintain this information so that the ACP node maintains its ACP address prefix. In EST renewal/rekeying, the ACP node's current ACP certificate is signaled during the TLS handshake.

This simple scenario has two limitations:

1. The ACP registrar 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 thereby create 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. 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 it 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 node's certificates, and therefore it is only necessary to revoke a compromised ACP registrar's 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 and/or revoked sub-CA certificate. Therefore, ACP domain membership for an ACP node enrolled by 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 candidate ACP members and attempt to get a new ACP certificate from an ACP registrar, for example, via BRSKI. As a result, ACP registrars that have an associated sub-CA make isolating and resolving issues with compromised registrars easier.

Note that ACP registrars with sub-CA functionality also can control the lifetime of ACP certificates more easily and therefore can be used as a tool to introduce short-lived certificates and to no longer 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 the following:

- Deciding which candidate ACP node is permitted or not permitted into an ACP domain. This may not be a decision to be made 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, potentially including a human decision in a NOC.
- Tracking all enrolled ACP nodes and their certificate information. For example, in support of revoking an individual ACP node's certificates.
- Needing more flexible policies as to which type of address prefix or even which 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 the ACP and/or the ANI

Both ACP and BRSKI require interfaces to be operational enough to support the sending and receiving of their packets. In node types where interfaces are enabled by default (e.g., without operator configuration), 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 disabled by default. In almost all network devices, though, it is common for configuration to change interfaces to a physically disabled state, and this would break the ACP.

In this section, we discuss a suggested operational model to enable and disable interfaces and nodes for ACP/ANI in a way that minimizes the risk of breaking the ACP due to operator action and also minimizes operator surprise when the 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 permitting the interface to send and receive packets necessary to operate ACP (or BRSKI) -- but not any other data plane packets. Unless the data plane is explicitly configured and enabled, all packets that are 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 mean the minimum signaling to auto-assign an IPv6 link-local address and talk to neighbors via their link-local addresses: SLAAC [RFC4862] and ND [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 supports 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. For example, "down" 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 are controlled via additional node-wide or interface-level options, but they all become active only 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 just a side effect in many cases, but it may be important in other cases (see Section 9.3.2.2).

A common problem of remote management is the operator or SDN controller cutting its own connectivity to the remote node via configuration, impacting its own management connection to the node. The ACP itself should have no dedicated configuration other than the aforementioned enabling of the ACP on brownfield ACP nodes. This leaves configuration that cannot distinguish between the 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 command that does this on many types of routers is the interface "down" command/configuration. When such a command is applied to the interface through which the ACP provides access for remote management, it cuts the remote management connection through the ACP because, as outlined above, the "down" command typically impacts the physical layer, too, and not only the data plane services.

To provide ACP/ANI resilience against such operator misconfiguration, this document recommends separating the "down" state of interfaces into an "admin down" state, where the physical layer is kept running and the 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 or received, packet filtering could be established automatically as described in Section 9.3.1.

An example of ANI, but not ACP, traffic that should be permitted to pass even in "admin down" state is BRSKI enrollment traffic between a BRSKI pledge and a BRSKI proxy.

New configuration options could be introduced as necessary (see discussion below) 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. Examples of checks include not allowing the option to be issued from a control connection (NETCONF/SSH) that goes across the interface itself ("do not disconnect yourself") or only applying the option after additional reconfirmation.

The following subsections discuss important aspects of the introduction of "admin down" state.

9.3.2.1. Security

Interfaces are physically brought down (or left in default "down" state) as a form of security. The "admin down" state as described above also provides also a high level of security because it only permits ACP/ANI operations, which are both well secured. Ultimately, it is subject to the deployment's security review 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 the following: 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 a user port. The device is disconnected from the network, and therefore diagnostics from the network side are no longer possible. Alternatively, all ports default to "admin down". The ACP (but not the data plane) would still automatically form. Diagnostics from the network side are possible, and operator reaction could include either to make this port the operational uplink port or to instruct re-cabling. Security wise, only the 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

The "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 and, today, never the result of autonomic L2/L3 services running on the nodes. Therefore, one option is to end the operator's reliance on interface state propagation via the subnet link or physical layer. This may not be possible when both sides are under the control of different operators, 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 autoconfigure a BFD session ("Bidirectional Forwarding Detection (BFD)" [RFC5880]) 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 means of diagnosing cabling issues in the absence of easier methods. It is more complex than automated neighbor diagnostics because it requires coordinated remote access to (likely) both 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 eliminate the need for "physical down" for neighbor diagnostics -- as long as both neighbors support ACP/ANI.

9.3.2.3. Low-Level Link Diagnostics

The "physical down" state is used to diagnose low-level interface behavior when higher-layer services (e.g., IPv6) are not working. Ethernet links are especially subject to a wide variety of possible incorrect configurations/cablings if they do not support automatic selection of variable parameters such as speed (10/100/1000 Mbps), crossover (automatic medium-dependent interface crossover (MDI-X)), and connector (fiber, copper -- when interfaces have multiple but can only enable one at a time). The need for low-level link diagnostics can therefore be minimized by using fully autoconfiguring links.

In addition to the "physical down" state, 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 the bringing down of all packet transmissions for reflection and/or cable-length measurements. Any of these options would disrupt ACP as well.

In cases where such low-level diagnostics of an operational link are desired but where the link could be a single point of failure for the ACP, the 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 types of interfaces inappropriate to operate purely for the ACP when they are not currently needed for the data plane.

9.3.3. Enabling Interface-Level ACP and ANI

The interface-level configuration option "ACP enable" enables ACP operations on an interface, starting with ACP neighbor discovery via DULL GRASP. 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, only "ANI enable" may need to

be supported and not "ACP enable". Unless overridden by global configuration options (see Section 9.3.4), either "ACP enable" or "ANI enable" (both abbreviated as "ACP/ANI enable") will result in the "down" state on an interface behaving 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, which allows more nodes of the ACP domain to be reached. 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 nonadjacent 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. As a 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 configuration including QoS, filtering, and other security on the data plane path across which this tunnel runs. This is the main reason 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 types 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 and/or configured.

In addition to the 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. Examples include DSL interfaces requiring Point-to-Point Protocol over Ethernet (PPPoE) credentials or mobile interfaces requiring credentials from a SIM card. Whatever mechanism is used to provide the necessary configuration to the device to enable the interface can also be expanded to decide 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 and/or 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 an erroneous "touch" is easily fixed (does not create 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. Enabling Node-Level ACP and ANI

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 Section 9.3.4.

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 the setup 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 (i.e., 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. The automatic enabling of ACP/ANI in networks where the operator does not want ACP/ANI or has 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 critical, security issue. If an attacker could impersonate the operator by registering as the operator at the MASA or otherwise getting hold of vouchers and could 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 enables the ANI, this could not happen because the operator would create a BRSKI registrar that would discover attack attempts, and the operator would set up his registrar with the MASA. Nodes requiring "ownership vouchers" would not be

subject to that attack. See [RFC8995] for more details. Note that a global "ACP enable" alone is not subject to these types of attacks because they always depend 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 and/or mechanism that will enroll the node with the ACP keying material: the ACP certificate and the TA. For ANI nodes, this protocol/mechanism is BRSKI.

When such a node is powered on and determines that it is in greenfield condition, it enables the bootstrap protocol(s) and/or mechanism(s). Once the ACP keying material is enrolled, the 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 when it is powered 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 and/or mechanisms.

The ACP greenfield state ends either through the successful enrollment of ACP keying material (certificate and TA) or the 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 and/or provisioning, especially when only a subset of nodes are expected to be deployed with ACP. Such an ACP node **SHOULD** observe attempts to provision or configure the node via interfaces and/or 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. This terminates ACP greenfield operations.

When an ACP greenfield node enables multiple, automated ACP or non-ACP enrollment and/or bootstrap protocols or mechanisms in parallel, care must be taken not to terminate any protocol/mechanism before the others either have succeeded in enrolling ACP keying material or have progressed to a point of permitted termination 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, "Opportunistic Security: Some Protection Most of the Time" [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 without authorization. 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 the "down" state as "admin down" and not "physical down". In the "admin-down" state, 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" commands 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" globally 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 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 disjoint ACP to which one or more unique Zone-IDs are assigned: ACP nodes registered for a specific ACP zone have to receive 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 TAs 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 L3 VPN PE (MPLS or IPv6 L3VPN), 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 VPN's non-autonomic routing protocol(s).

While such a setup is possible with any ACP addressing sub-scheme, the Zone Addressing Sub-Scheme makes it easy to configure and scalable for any VPN routing protocols because every ACP zone only needs 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 requires 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 nonstandard enhancements, for example, by allowing GRASP messages to be propagated across the L3VPN, leveraging for example L3VPN multicast support.

Finally, these partial deployments can be merged into a single, contiguous ACP that is completely autonomous (given appropriate ACP support across the core) without changes in the cryptographic 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 preexisting mechanisms for configuration such as CLI or YANG data models ("The YANG 1.1 Data Modeling Language" [RFC7950]).

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 a certificate needs to be configured via some preexisting mechanisms outside the scope of this specification. Today, routers typically have 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 preexisting 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 the 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 auto-discover 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 done more reliably across the ACP itself because the ACP provides addressing and connectivity (routing) independent of the data plane. For this, the configuration methods simply need to allow operating across the ACP VRF, for example, with 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 (for example, SNMP, TFTP, or 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 of the configuration taking place (for example, transport flows, port numbers, and/or IP addresses).

The ACP can and should be used equally 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 the 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 on 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 for ACP implementations to require this enhancement, though, in order 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 a few central dependencies: a CRL may not be available during a network partition. This can be addressed by a suitable policy to not immediately disconnect neighbors when no CRL is available. Also, an ACP registrar or CA 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-CAs, 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 merging of partitions will not cause problems with ACP addresses assigned during partitioning.

After a network partition, merging 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, the merging will be smooth.

Merging two networks with different TAs requires the ACP nodes to trust the union of TAs. As long as the routing-subdomain hashes are different, the addressing will not overlap. Overlaps will only happen accidentally in the unlikely event of a 40-bit hash collision in SHA-256 (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 an 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 administrative down 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 inside the ACP, and also provide replay protection.

An attacker will not be able to join the ACP unless it has a valid ACP certificate. An on-path attacker without a valid ACP certificate cannot inject packets into the ACP due to ACP secure channels. An attacker also cannot decrypt ACP traffic unless it can crack the encryption. It 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 the configuring of non-autonomic ACP options if those are supported on the impaired ACP nodes (see Section 8). Injecting traffic into or extracting traffic 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 node's 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 due to some vendor, product, or customer limitations. This includes protocols such as SNMP ("An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks" [RFC3411]), NTP [RFC5905], PTP (Precision Time Protocol [IEEE-1588-2008]), DNS ("DNS Extensions to Support IP Version 6" [RFC3596]), DHCPv6 ("Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" [RFC3315]), syslog ("The BSD Syslog Protocol" [RFC3164]), RADIUS ("Remote Authentication Dial In User Service (RADIUS)" [RFC2865]), Diameter ("Diameter Base Protocol" [RFC6733]), TACACS ("An Access Control Protocol, Sometimes Called TACACS" [RFC1492]), IPFIX ("Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information" [RFC7011]), NetFlow ("Cisco Systems NetFlow Services Export Version 9" [RFC3954]) -- just to name a few. Not all of these protocol references are necessarily the latest version of protocols, but they are 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 to 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 greater physical protection. The ULA addresses are only reachable inside the ACP context and therefore unreachable from the data plane. Also, the ACP protocols should be implemented to be attack resistant and to 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 be reachable only from the ACP and not from the data plane. This applies especially to those management plane functions that lack secure end-to-end transport and to which the ACP provides both automatic, reliable connectivity and protection against attacks. Protection across all potential attack vectors is typically easier to do in devices whose software is designed from the beginning with the ACP in mind than in 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, but they can just filter them (which is unavoidable by any means).

See Appendix A.9.8 for further considerations on avoiding and dealing with compromised nodes.

10.3. The Administrator View

An ACP is self-forming, self-managing, and self-protecting; therefore, it 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 for 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 into the ACP and all its parameters to be able to troubleshoot. Therefore, an ACP must support all show and debug options, as with any other network function. Specifically, an NMS or controller must be able to discover the ACP and monitor its health. This visibility into ACP operations must clearly be separated from the visibility of the 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 that does not support the ACP or that does not have a valid domain certificate cannot connect to it. This means that by default a traditional controller or NMS cannot connect to an ACP. See Section 8.1.1 for details on how to connect an NMS host to 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 the requirements for 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 addresses 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.509 v3 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 its IDevID is assigned, the node needs (1) to utilize hardware support such as a Trusted Platform Module (TPM) to protect against extraction and/or cloning of the private key of the IDevID and (2) a hardware/software infrastructure to prohibit execution of unauthenticated 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 and/or 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. The 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 that supports 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 by making ACP nodes attempt their ACP certificate renewal via this impaired ACP node. This problem can be avoided when the EST server implementation can verify that the configured CA is indeed providing renewal for certificates of the node's ACP. The ability to do so depends on the protocol between the EST server and the CA, 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 and/or software components, including options such as TPM for IDevID and/or ACP certificate, prohibitions against the execution of untrusted software, and design aspects of the EST server functionality for the ACP that 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.

The security model of the ACP as defined in this document is tailored for use with private PKI. The TA of a private PKI provides the security against maliciously created ACP certificates that 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).

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 protocols and/or services running 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 the management of future autonomic peer-to-peer/distributed networks. Any ACP member can send ACP IPv6 packets to other ACP members and announce via ACP GRASP services to all ACP members without depending on 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, 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 or protocols using the ACP. Because no group keys are used, there is no risk of impacted nodes accessing 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 to change or 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 nonrenewal of short-lived certificates. In this specification, there are no further optimizations 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 or more secure. For example, central network configuration relies on a security model where only a 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 introducing roles. See Appendix A.9.5.

Operators and developers of provisioning software need to be aware of how the provisioning and configuration of network devices impacts the ability of the operator and/or provisioning software to remotely access the network nodes. By using the ACP, most of the issues of provisioning/configuration causing connectivity loss of remote provisioning and configuration will be eliminated, see Section 6. Only a few exceptions, such as explicit physical interface down configuration, will be left. See 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 also by 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, autoconfiguring 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 on any non-ACP (data plane) operations and/or 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 reenrolled even after unintentional expiry (e.g., due to 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 a much higher likelihood of attacks by other nodes "stealing" L2 addresses than in the actual routed case, especially when the bridged network includes untrusted 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 Neighbor Discovery Protocol (NDP) or DHCP learning which port/MAC-address and IPv6 address belong together and blocking 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 has assigned value 97 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 has assigned value 10 for id-on-AcpNodeName in the "SMI Security for PKIX Other Name Forms" (1.3.6.1.5.5.7.8) registry.

IANA has registered the names in Table 2 in the "GRASP Objective Names" subregistry of the "GeneRic Autonomic Signaling Protocol (GRASP) Parameters" registry.

Objective Name	Reference
AN_ACP	RFC 8994 (Section 6.4)
SRV.est	RFC 8994 (Section 6.2.5)

Table 2: GRASP Objective Names

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 by 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.

IANA has created an "Autonomic Control Plane (ACP)" registry with the subregistry, "ACP Address Type" (Table 3).

Value	Description	Reference
0	ACP Zone Addressing Sub-Scheme/ACP Manual Addressing Sub-Scheme	RFC 8994 (Section 6.11.3, Section 6.11.4)
1	ACP Vlong Addressing Sub-Scheme	RFC 8994 (Section 6.11.5)
2-3	Unassigned	

Table 3: Initial Values in the "ACP Address Type" Subregistry

The values in the "ACP Address Type" subregistry are numeric values 0..3 paired with a name (string). Future values **MUST** be assigned using the Standards Action policy defined by "Guidelines for Writing an IANA Considerations Section in RFCs" [RFC8126].

13. References

13.1. Normative References

[IKEV2IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <https://www.iana.org/assignments/ikev2-parameters>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[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>.

[RFC5954] Gurbani, V., Ed., Carpenter, B., Ed., and B. Tate, Ed., "Essential Correction for IPv6 ABNF and URI Comparison in RFC 3261", RFC 5954, DOI 10.17487/RFC5954, August 2010, <https://www.rfc-editor.org/info/rfc5954>.

[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>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

13.2. Informative References

[AR8021] IEEE, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", IEEE 802.1AR, <https://1.ieee802.org/security/802-1ar>.

[CABFORUM] CA/Browser Forum, "Certificate Contents for Baseline SSL", November 2019, <https://cabforum.org/baseline-requirements-certificate-contents/>.

[FCC] FCC, "June 15, 2020 T-Mobile Network Outage Report", A Report of the Public Safety and Homeland Security Bureau Federal Communications Commission, PS Docket No. 20-183, October 2020, <https://docs.fcc.gov/public/attachments/DOC-367699A1.docx>.

[IEEE-1588-2008] IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", DOI 10.1109/IEEESTD.2008.4579760, IEEE 1588-2008, July 2008, <https://standards.ieee.org/standard/1588-2008.html>.

[IEEE-802.1X] IEEE, "IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", DOI 10.1109/IEEESTD.2010.5409813, IEEE 802.1X-2010, February 2010, <https://standards.ieee.org/standard/802_1X-2010.html>.

[LLDP] IEEE, "IEEE Standard for Local and metropolitan area networks: Station and Media Access Control Connectivity Discovery", DOI 10.1109/IEEESTD.2016.7433915, IEEE 802.1AB-2016, March 2016, <https://standards.ieee.org/standard/802_1AB-2016.html>.

[MACSEC] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security", DOI 10.1109/IEEESTD.2006.245590, IEEE 802.1AE-2006, August 2006, <https://standards.ieee.org/standard/802_1AE-2006.html>.

[NOC-AUTOCONFIG] Eckert, T., Ed., "Autoconfiguration of NOC services in ACP networks via GRASP", Work in Progress, Internet-Draft, draft-eckert-anima-noc-autoconfig-00, 2 July 2018, <https://tools.ietf.org/html/draft-eckert-anima-noc-autoconfig-00>.

[OP-TECH] Wikipedia, "Operational technology", October 2020, <https://en.wikipedia.org/w/index.php?title=Operational_technology&oldid=986363045>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC1492] Finseth, C., "An Access Control Protocol, Sometimes Called TACACS", RFC 1492, DOI 10.17487/RFC1492, July 1993, <https://www.rfc-editor.org/info/rfc1492>.

[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>.

[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>.

[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>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[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>.

[RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/info/rfc8684>.

[RFC8739] Sheffer, Y., Lopez, D., Gonzalez de Dios, O., Pastor Perales, A., and T. Fossati, "Support for Short-Term, Automatically Renewed (STAR) Certificates in the Automated Certificate Management Environment (ACME)", RFC 8739, DOI 10.17487/RFC8739, March 2020, <https://www.rfc-editor.org/info/rfc8739>.

[RFC8981] Gont, F., Krishnan, S., Narten, T., and R. Draves, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6", RFC 8981, DOI 10.17487/RFC8981, February 2021, <https://www.rfc-editor.org/info/rfc8981>.

[RFC8992] Jiang, S., Ed., Du, Z., Carpenter, B., and Q. Sun, "Autonomic IPv6 Edge Prefix Management in Large-Scale Networks", RFC 8992, DOI 10.17487/RFC8992, May 2021, <https://www.rfc-editor.org/info/rfc8992>.

[RFC8993] Behringer, M., Ed., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", RFC 8993, DOI 10.17487/RFC8993, May 2021, <https://www.rfc-editor.org/info/rfc8993>.

[ROLL-APPLICABILITY] Richardson, M., "ROLL Applicability Statement Template", Work in Progress, Internet-Draft, draft-ietf-roll-applicability-template-09, 3 May 2016, <https://tools.ietf.org/html/draft-ietf-roll-applicability-template-09>.

[SR] Wikipedia, "Single-root input/output virtualization", September 2020, <https://en.wikipedia.org/w/index.php?title=Single-root_input/output_virtualization&oldid=978867619>.

[TLS-DTLS13] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <https://tools.ietf.org/html/draft-ietf-tls-dtls13-43>.

[X.509] ITU-T, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, October 2016, <https://www.itu.int/rec/T-REC-X.509>.

[X.520] ITU-T, "Information technology - Open Systems Interconnection - The Directory: Selected attribute types", ITU-T Recommendation X.520, October 2016, <https://www.itu.int/rec/T-REC-X.520>.

Appendix A. Background and Future (Informative)

The following sections provide 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 discuss possible future variations of the ACP.

A.1. ACP Address Space Schemes

This document defines the Zone, Vlong, and Manual Addressing Sub-Schemes primarily to support address prefix assignment via distributed, potentially uncoordinated ACP registrars as defined in Section 6.11.7. This costs a 48/46-bit identifier so that these ACP registrars can assign nonconflicting 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. As a result, the Zone and Vlong 8/16 Addressing Sub-Schemes attempt to support all features but via separate prefixes.

In networks that expect always to rely on a centralized PMS as described 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-scheme 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 non-autonomic configuration of any non-"Manual" ACP address spaces and therefore ensure that such non-autonomic 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, and/or server setup). This includes otherwise unconfigured 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 [RFC8993]. 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, reduced-security 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 a pledge) via its response to the pledge's EST CSR Attributes Request that is mandatory in BRSKI.

The CA in an ACP network must not change the otherName / AcpNodeName in the certificate. The ACP nodes can therefore find their ACP addresses 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, on 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 the BRSKI registrar/CA.

In the absence of BRSKI or less secure variants thereof, the provisioning of certificates may involve one or more touches or nonstandardized automation. Node vendors usually support the provisioning of certificates into nodes via PKCS #7 (see "PKCS #7: Cryptographic Message Syntax Version 1.5" [RFC2315]) and may support this provisioning through vendor-specific models via NETCONF ("Network Configuration Protocol (NETCONF)" [RFC6241]). If such nodes also support NETCONF Zero Touch [RFC8572], then this can be combined with zero-touch provisioning of domain certificates into nodes. Unless there is the equivalent integration of NETCONF connections across the ACP as there is in BRSKI, this combination would not support zero-touch bootstrap across an unconfigured 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 that were considered were GRASP, mDNS, and LLDP.

A.3.1. LLDP

LLDP and Cisco's earlier Cisco Discovery Protocol (CDP) are examples of L2 discovery protocols that terminate their messages on L2 ports. If those protocols had been chosen for ACP neighbor discovery, ACP neighbor discovery would also have terminated on L2 ports. This would have prevented ACP construction over non-ACP-capable, but LLDP- or CDP-enabled L2 switches. LLDP has extensions using different MAC addresses, and this could have been an option for ACP discovery as well, but the additional required IEEE standardization and definition of a profile for such a modified instance of LLDP seemed to be more work than the benefit of "reusing the existing protocol" LLDP for this very simple purpose.

A.3.2. mDNS and L2 Support

Multicast DNS (mDNS) "Multicast DNS" [RFC6762] with DNS Service Discovery (DNS-SD) Resource Records (RRs) as defined in "DNS-Based Service Discovery" [RFC6763] was a key contender as an ACP discovery protocol. Because it relies on link-local IP multicast, it operates at the subnet level and is also found in L2 switches. The authors of this document are not aware of an mDNS implementation that terminates its mDNS messages on L2 ports instead of on 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 to 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 were derived from a pre-standard implementation of ACP that was successfully deployed in operational networks.

The requirements for routing in the ACP are the following:

- Self-management: the ACP must build automatically, without human intervention. Therefore, the 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 requirements, both in terms of memory and CPU. 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 ANI, node addresses are identifiers, they 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 for more complex methods later, 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 ANI is a new concept, it is likely that changes to 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 delay-insensitive feedback loops. It is possible that different optimization schemes will become necessary in the future, but RPL can be expanded (see "Extensibility" above).

A.5. ACP Information Distribution and Multicast

IP multicast is not used by the ACP because the ANI 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 autoconfiguring solution for ASM (Any Source Multicast - the original form of IP multicast defined in "Host extensions for IP multicasting" [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 "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)" [RFC7761]). The other aspects of complexity are the implementation of MLD ("Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast" [RFC4604]), PIM-SM, and Anycast-RP (see "Anycast-RP Using Protocol Independent Multicast (PIM)" [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 turn 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 (see "Source-Specific Multicast for IP" [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 configuration other than just enabling it on all nodes, and it only requires a simpler version of MLD (see "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols" [RFC5790]).

IGP routing protocols based on LSP (Link State Protocol) 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-based 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 expect that many future services in the ASA will have only a few consuming ASAs, and for those cases, the M_DISCOVERY method is more efficient than flooding across the whole domain.

Because the ACP uses RPL, one desirable future extension is to use RPL's 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 for 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 solutions 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 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 the creation of multiple physical networks that initially may be separated with one ACP domain but different routing subdomains, so that all nodes can mutually 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 ACP domains need to be created that are not allowed to connect to each other by default, simply use 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 include the comparison of the domain part.

If multiple, independent networks chose the same domain name but had their own CAs, 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 have a high probability of being unique. It is recommended to use domain names that start with a DNS domain name 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 [RFC8993] 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): ACP nodes with the domains "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 domain's 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 connections 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 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 types 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 the derivation of its prefix from the domain name so that no address allocation is required to deploy the ACP. The ACP will equally work using any other /48 IPv6 prefix and not ULA. 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 operates 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 ACP address used by a device is derived from some preexisting, permanent local ID (such as MAC address), then it would be useful to store that local ID also in the certificate.

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 functions that impact transit connectivity 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 "Autonomic IPv6 Edge Prefix Management in Large-Scale Networks" [RFC8992].

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, which would 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. In future ACP extensions meant to better support constrained devices, end-to-end GRASP connections can be made to select their transport protocol by indicating the supported transport protocols (e.g. TLS/DTLS) via GRASP parameters of the GRASP objective through which the transport endpoint is discovered.

RPL, the routing protocol used for the ACP, explicitly does 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 which 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 that certificates would need to be updated, or it should be ensured 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. By leveraging the domain certificates, it would not be dependent on security provided 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 the 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 the /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 the full isolation of 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 [SR]) 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 not 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 uses the Extensible Authentication Protocol over Local Area Network (EAPoL) Ethertype (0x88A2).

Note that in the case of ANI nodes, all of 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 a YANG data model [RFC7950] and/or node-internal APIs to monitor and manage the ACP.

Support for the ACP adjacency table (Section 6.3) and ACP GRASP needs to be included in such model and/or API.

A.9.4. RPL Enhancements

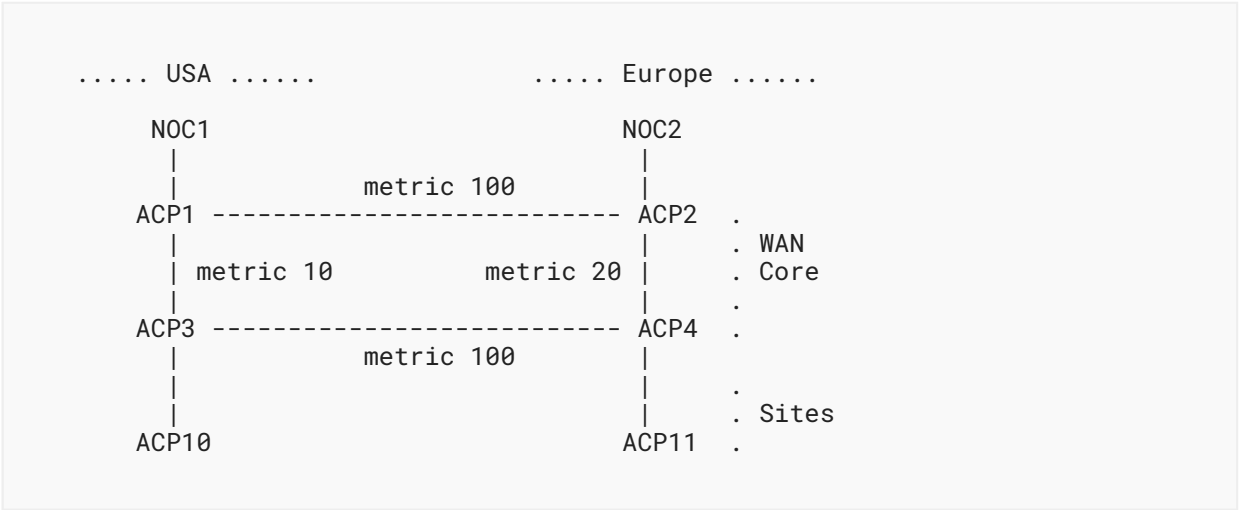

Figure 17: Dual NOC

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 17 shows an extreme example. Assuming that node ACP1 becomes the RPL root, traffic between ACP11 and NOC2 will pass through ACP4-ACP3-ACP1-ACP2 instead of ACP4-ACP2 because the RPL-calculated DODAG and routes are shortest paths towards the RPL root.

To overcome these limitations, extensions and/or modifications to the RPL profile can optimize for multiple NOCs. This requires utilizing data plane artifacts, including IP-in-IP encapsulation/decapsulation 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 certificate's 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 it is still subject to code modifications.

Another interesting role to assign to certificates is that of a NOC node. This would allow the limiting of certain types of connections, such as OAM TLS connections to only the NOC initiators or responders.

A.9.6. Autonomic L3 Transit

In this specification, the ACP can only establish autonomic connectivity across L2 hops but requires non-autonomic configuration to tunnel across L3 paths. Future work should specify mechanisms to automatically tunnel ACP across L3 networks. A hub-and-spoke option would allow tunneling 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 applied without changing the external, interoperability-affecting characteristics of ACP implementations.

Even better diagnostics of ACP operations are possible with additional signaling extensions, such as the following:

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). This includes potentially new information elements.
2. As an 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 (the 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 types of compromise are the leakage of credentials to manage and/or 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 types 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 the change of these authentication mechanisms, but ideally they should be autoconfiguring across the ACP. See [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 configuration unless factory reset is done.

With such means, the valid administration has the best chances to maintain access to ACP nodes, to discover malicious configuration though ongoing configuration tracking from central locations, for example, and to react accordingly.

The primary reaction is to withdraw or change credentials, terminate malicious existing management sessions, and fix the configuration. Ensuring that management sessions using invalidated credentials are terminated automatically without recourse will likely require new work.

Only when these steps are infeasible, 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 the GRASP flooding distribution of triggered updates of CRL/OCSP or the explicit removal indication of the compromised node's 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 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 and/or modifying DULL GRASP messages and/or actual secure channel data packets. For example, if at some point in time, DTLS traffic could be more easily decrypted than traffic of IKEv2, the MITM could filter all IKEv2 packets to force ACP nodes to use DTLS (assuming that the ACP nodes in question supported both DTLS and IKEv2).

For cases where such MITM attacks are not capable of injecting malicious traffic (but only of decrypting the traffic), a downgrade attack could be discovered after a secure channel connection is established, for example, by using 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 signal the ACP secure channel options announced by DULL GRASP 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, i.e., the ACP peer that decides which secure channel protocol to use. 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 (e.g., a 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.

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 whom (in alphabetical order): Ignas Bagdonas, Parag Bhide, Balaji BL, Alex Clemm, Yves Hertoghs, Bruno Klauser, Max Pritikin, Michael Richardson, and Ravi Kumar Vadapalli.

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

Many thanks to Ben Kaduk, Roman Danyliw, 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 to Carsten Bormann for CBOR/CDDL help.

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

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

Authors' Addresses

Toerless Eckert (EDITOR)
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara, CA 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 of America
Email: sbjarnason@arbor.net