ANIMA WG   T. Eckert, Ed.
Internet-Draft   Futurewei USA
Intended status: Standards Track   M. Behringer, Ed.
Expires: 3 May 2021

                                                           S. Bjarnason
                                                          Arbor Networks
                                                         30 October 2020

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

Abstract

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

Table of Contents

1. Introduction (Informative)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.1. Applicability and Scope

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

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

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

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

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

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

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

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

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

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

2. Acronyms and Terminology (Informative)

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

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

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

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

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

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

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

ACP address: An IPv6 address assigned to the ACP node. It is stored
   in the acp-node-name of the ->"ACP certificate".

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

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

ACP domain: The ACP domain is the set of nodes with ->"ACP
   certificates" that allow them to authenticate each other as
   members of the ACP domain. See also Section 6.2.3.

ACP (ANI/AN) certificate: A [RFC5280] certificate (LDevID) carrying
   the acp-node-name which is used by the ACP to learn its address in
   the ACP and to derive and cryptographically assert its membership
   in the ACP domain.

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

ACP Loopback interface: The Loopback interface in the ACP Virtual
   Routing and Forwarding (VRF) that has the ACP address assigned to
   it. See Section 6.13.5.1.

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

ACP (ULA) prefix(es): The /48 IPv6 address prefixes used across the
   ACP. In the normal/simple case, the ACP has one ULA prefix, see
   Section 6.11. The ACP routing table may include multiple ULA
   prefixes if the "rsub" option is used to create addresses from
   more than one ULA prefix. See Section 6.2.2. The ACP may also
   include non-ULA prefixes if those are configured on ACP connect
   interfaces. See Section 8.1.1.

ACP secure channel: A channel authenticated via ->"ACP certificates"

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

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

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

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

(AN) Domain Name: An FQDN (Fully Qualified Domain Name) in the acp-
node-name of the Domain Certificate. See Section 6.2.2.

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

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

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

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

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

CA: "Certification Authority". An entity that issues digital
certificates. A CA uses its private key to sign the certificates
it issues. Relying parties use the public key in the CA
certificate to validate the signature.

CRL: "Certificate Revocation List". A list of revoked certificates.
Required to revoke certificates before their lifetime expires.

Data-Plane: The counterpoint to the ACP VRF in an ACP node:
forwarding of user traffic and in non-autonomous nodes/networks
also any non-autonomous control and/or management plane functions.
In a fully Autonomic Network node, the Data-Plane is managed
autonomically via Autonomic Functions and Intent. See Section 1
for more detailed explanations.

device: A physical system, or physical node.

Enrollment: The process through which a node authenticates itself to
   a network with an initial identity, which is often called IDevID
   certificate, and acquires from the network a network specific
   identity, which is often called LDevID certificate, and
   certificates of one or more Trust Anchor(s). In the ACP, the
   LDevID certificate is called the ACP certificate.
EST: "Enrollment over Secure Transport" ([RFC7030]). IETF standard-
   track protocol for enrollment of a node with an LDevID
   certificate. BRSKI is based on EST.
GRASP: "Generic Autonomic Signaling Protocol". An extensible
   signaling protocol required by the ACP for ACP neighbor discovery.
   The ACP also provides the "security and transport substrate" for
   the "ACP instance of GRASP". This instance of GRASP runs across
   the ACP secure channels to support BRSKI and other NOC/OAM or
   Autonomic Functions. See [I-D.ietf-anima-grasp].
IDevID: An "Initial Device IDentity" X.509 certificate installed by
   the vendor on new equipment. Contains information that
   establishes the identity of the node in the context of its vendor/
   manufacturer such as device model/type and serial number. See
   [AR8021]. The IDevID certificate cannot be used as a node
   identifier for the ACP because they are not provisioned by the
   owner of the network, so they can not directly indicate an ACP
   domain they belong to.
in-band (management/signaling): In-band management traffic and/or
   control plane signaling uses the same network resources such as
   routers/switches and network links that it manages/controls. In-
   band is the standard management and signaling mechanism in IP
   networks. Compared to ->"out-of-band" it requires no additional
   physical resources, but introduces potentially circular
   dependencies for its correct operations. See ->"introduction".
Intent: Policy language of an autonomic network according to
   [I-D.ietf-anima-reference-model].
Loopback interface: See ->"ACP Loopback interface".
LDevID: A "Local Device IDentity" is an X.509 certificate installed
   during "enrollment". The Domain Certificate used by the ACP is an
   LDevID certificate. See [AR8021].
Management: Used in this document as another word for ->"OAM".
MASA (service): "Manufacturer Authorized Signing Authority". A
   vendor/manufacturer or delegated cloud service on the Internet
   used as part of the BRSKI protocol.
MIC: "Manufacturer Installed Certificate". This is another word to
   describe an IDevID in referenced materials. This term is not used
   in this document.
native interface: Interfaces existing on a node without
   configuration of the already running node. On physical nodes
   these are usually physical interfaces; on virtual nodes their
   equivalent.
NOC: Network Operations Center.

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

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

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

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

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

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

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

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

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

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

3.1. An Infrastructure for Autonomic Functions

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

3.2. Secure Bootstrap over a not configured Network

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

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

3.3. Data-Plane Independent Permanent Reachability

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

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

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

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

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

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

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

4. Requirements (Informative)

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

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

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

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

5. Overview (Informative)

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

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

   Note:

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

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

   The following figure illustrates the ACP.

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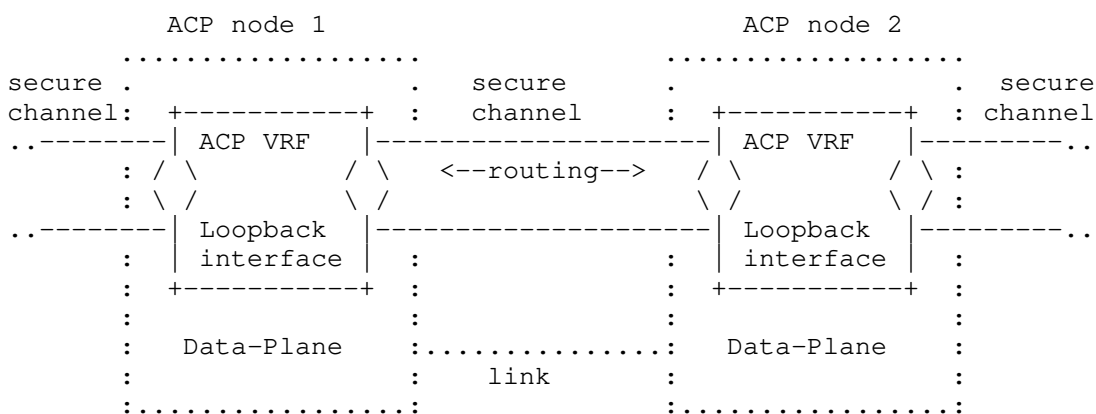
                  Figure 1: ACP VRF and secure channels

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

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

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

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

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

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

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

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

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

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

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

6.2. ACP Domain, Certificate and Network

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

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

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

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

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

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

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

6.2.1. ACP Certificates

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.2.2. ACP Certificate AcpNodeName

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

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

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

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

Figure 2: ACP Node Name ABNF

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

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

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

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

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

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

The following points explain and justify the encoding choices described:

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

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

3.2 The element should not require additional ASN.1 en/decoding,
because libraries to access certificate information
especially for embedded devices may not support extended
ASN.1 decoding beyond predefined, mandatory fields.
subjectAltName / otherName is already used with a single
string parameter for several otherNames (see [RFC3920],
[RFC7585], [RFC4985], [RFC8398]).

3.3 The element required for the ACP should minimize the risk of
being misinterpreted by other uses of the LDevID
certificate. It also must not be misinterpreted to actually
be an email address, hence the use of the otherName /
rfc822Name option in the certificate would be inappropriate.

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

6.2.2.1. AcpNodeName ASN.1 Module

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

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

DEFINITIONS IMPLICIT TAGS ::=
BEGIN

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

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

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

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

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

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

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

END
```

                            Figure 3

6.2.3. ACP domain membership check

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

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

2: The peer's certificate passes certificate path validation as
    defined in [RFC5280], section 6 against one of the TA associated
    with the ACP node's ACP certificate (see Section 6.2.4 below).
    This includes verification of the validity (lifetime) of the
    certificates in the path.

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

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

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

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

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

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

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

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

In summary:

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

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

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

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

6.2.3.1. Realtime clock and Time Validation

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

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

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

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

6.2.4. Trust Anchors (TA)

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

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

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

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

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

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

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

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

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

6.2.5. Certificate and Trust Anchor Maintenance

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

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

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

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

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

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

6.2.5.1. GRASP objective for EST server

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

     Example:

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

                    Figure 4: GRASP SRV.est example

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

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

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

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

Figure 5: GRASP SRV.est definition

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

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

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

6.2.5.2. Renewal

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

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

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

6.2.5.3. Certificate Revocation Lists (CRLs)

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

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

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

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

6.2.5.4. Lifetimes

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

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

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

6.2.5.5. Re-enrollment

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

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

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

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

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

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

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

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

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

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

6.2.5.6. Failing Certificates

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

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

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

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

6.3. ACP Adjacency Table

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

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

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

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

6.4. Neighbor Discovery with DULL GRASP

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

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

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

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

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

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

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

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

                    Figure 6: GRASP AN_ACP example

   The formal CDDL definition is:

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

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

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

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

                  Figure 7: GRASP AN_ACP definition

   The objective-flags field is set to indicate synchronization.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.5. Candidate ACP Neighbor Selection

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

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

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

6.6. Channel Selection

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

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

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

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

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

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

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

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

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

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

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

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

    [3]   Node 2 receives [1] from Node 1

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

    [5]   Node 1 receives [2] from Node 2

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

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

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

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

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

    [11:C2] Node 1 certificate has lower ACP Node-ID than Node2,
            therefore Node 1 considers itself the Follower and Node 2 the
            Decider on connection C2, but they also identify that C2 is
            to the same mutual peer as their C1, so this has no further
            impact: the roles Decider and Follower where already assigned
            between these two peers by [8:C1].

    [12:C2] Node 2 (the Decider) closes C1. Node 1 is fine with this,
            because of its role as the Follower (from [8:C1]).

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

                    Figure 8: Secure Channel sequence of steps

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

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

6.7. Candidate ACP Neighbor verification

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

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

6.8. Security Association (Secure Channel) protocols

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

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

6.8.1. General considerations

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

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

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

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

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

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

6.8.2. Common requirements

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

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

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

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

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

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

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

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

6.8.3. ACP via IPsec

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

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

6.8.3.1. Native IPsec

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

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

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

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

6.8.3.1.1. RFC8221 (IPsec/ESP)

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

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

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

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

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

Explanations:

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

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

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

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

6.8.3.1.2. RFC8247 (IKEv2)

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

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

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

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

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

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

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

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

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

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

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

6.8.3.2. IPsec with GRE encapsulation

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

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

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

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

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

6.8.4. ACP via DTLS

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

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

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

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

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

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

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

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

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

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

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

6.8.5. ACP Secure Channel Profiles

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

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

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

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

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

6.9. GRASP in the ACP

6.9.1. GRASP as a core service of the ACP

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

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

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

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

6.9.2. ACP as the Security and Transport substrate for GRASP

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

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

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

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

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

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

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

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

              Figure 9: ACP as security and transport substrate for GRASP

6.9.2.1. Discussion

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

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

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

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

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

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

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

6.10. Context Separation

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

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

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

6.11. Addressing inside the ACP

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

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

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

6.11.1. Fundamental Concepts of Autonomic Addressing

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

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

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

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

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

6.11.2. The ACP Addressing Base Scheme

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

```
  8       40                        2                  78
  +--+-----------------------+------+---------------------------+
  |fd| hash(routing-subdomain) | Type |        (sub-scheme)       |
  +--+-----------------------+------+---------------------------+
```
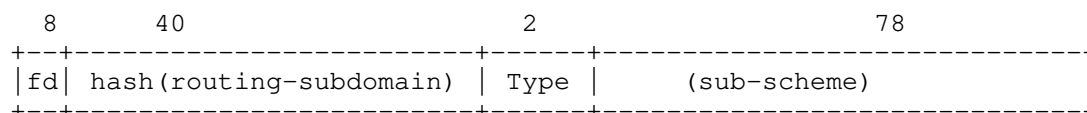
                  Figure 10: ACP Addressing Base Scheme

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

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

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

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

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

   The following summarizes the addressing Sub-Schemes:

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

                   Figure 11: Addressing Sub-Schemes

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

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

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

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

                Figure 12: ACP Zone Addressing Sub-Scheme

   The fields are defined as follows:

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

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

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

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

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

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

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

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

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

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

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

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

```
              64                                    64
+--------------------+---+----------++----------------------------+
|   (base scheme)    | Z | Subnet-ID||    Interface Identifier    |
+--------------------+---+----------++----------------------------+
         50            1      13
```
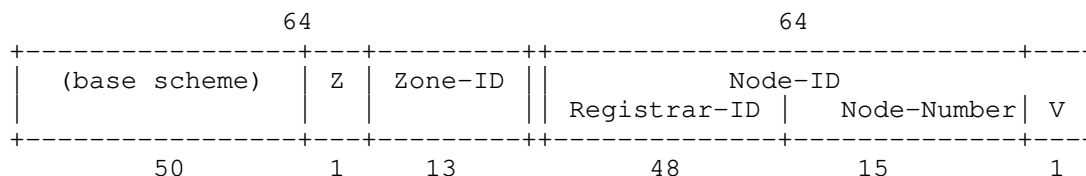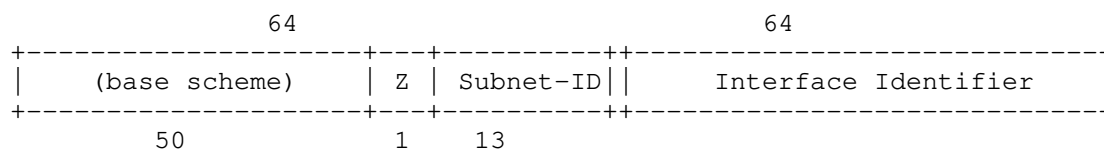
Figure 13: ACP Manual Addressing Sub-Scheme

The fields are defined as follows:

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

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

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

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

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

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

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

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

```
             50                               78
+--------------------++----------------------------+----------+
|                    ||              Node-ID        |          |
|   (base scheme)    ||  Registrar-ID |F| Node-Number|     V    |
+--------------------++-------------+-------------+----------+
         50               46          1    23/15      8/16
```
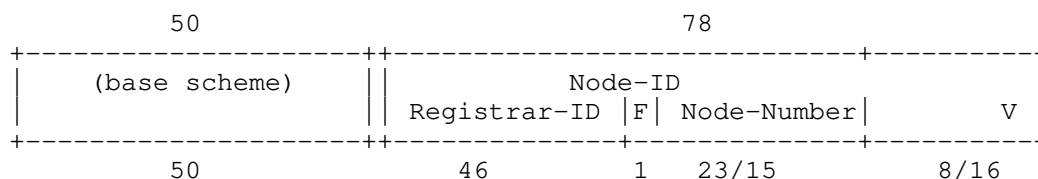
Figure 14: ACP Vlong Addressing Sub-Scheme

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

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

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

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

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

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

6.11.6. Other ACP Addressing Sub-Schemes

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

6.11.7. ACP Registrars

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

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

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

6.11.7.1. Use of BRSKI or other Mechanism/Protocols

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

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

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

6.11.7.2. Unique Address/Prefix allocation

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

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

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

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

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

6.11.7.3. Addressing Sub-Scheme Policies

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

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

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

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

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

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

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

6.11.7.4. Address/Prefix Persistence

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

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

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

6.11.7.5. Further Details

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

6.12. Routing in the ACP

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

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

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

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

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

6.12.1. ACP RPL Profile

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

6.12.1.1. Overview

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

6.12.1.1.1. Single Instance

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

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

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

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

6.12.1.1.2. Reconvergence

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

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

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

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

6.12.1.2. RPL Instances

   Single RPL instance. Default RPLInstanceID = 0.

6.12.1.3. Storing vs. Non-Storing Mode

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

6.12.1.4. DAO Policy

   Proactive, aggressive DAO state maintenance:

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

6.12.1.5. Path Metric

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

6.12.1.6. Objective Function

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

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

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

6.12.1.7. DODAG Repair

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

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

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

stretch_rank: none provided ("not stretched").

Data Path Validation: Not used.

Trickle: Not used.

6.12.1.8. Multicast

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

6.12.1.9. Security

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

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

6.12.1.10. P2P communications

Not used.

6.12.1.11. IPv6 address configuration

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

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

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

6.12.1.12. Administrative parameters

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

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

6.12.1.13. RPL Packet Information

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

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

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

6.12.1.14. Unknown Destinations

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

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

6.13. General ACP Considerations

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

6.13.1. Performance

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

6.13.2. Addressing of Secure Channels

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

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

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

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

6.13.3. MTU

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

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

6.13.4. Multiple links between nodes

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

6.13.5. ACP interfaces

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

6.13.5.1. ACP loopback interfaces

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

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

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

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

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

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

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

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

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

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

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

6.13.5.2. ACP virtual interfaces

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

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

6.13.5.2.1. ACP point-to-point virtual interfaces

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

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

6.13.5.2.2. ACP multi-access virtual interfaces

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.1. Why (Benefits of ACP on L2 switches)

```
ANrtr1 ------ ANswitch1 --- ANswitch2 ------- ANrtr2
       .../     \                     \  ...
ANrtrM ------     \                     ------- ANrtrN
                   ANswitchM ...
```
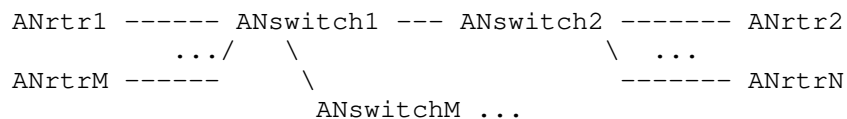
Figure 15: Topology with L2 ACP switches

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

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

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

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

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

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

7.2. How (per L2 port DULL GRASP)

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

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

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

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

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

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

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

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

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

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

8. Support for Non-ACP Components (Normative)

8.1. ACP Connect

8.1.1. Non-ACP Controller / NMS system

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

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

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

```
                                      Data-Plane "native" (no ACP)
                                        .
+--------+        +----------------+    .        +-------------+
|  ACP   |        | ACP Edge Node  |    .        |             |
|  Node  |        |                |    v        |             |
|        |--------|...[ACP VRF]....+---------------|             |+
|        |     ^  | .              |             |  NOC Device ||
|        |     .  |  .[Data-Plane]..+---------------| "NMS hosts" ||
|        |     .  |  [          ] | .        ^    |             ||
+--------+     .  +----------------+ .        .    +-------------+|
              .                      .        .     +------------+
              .                      .        .
        Data-Plane "native"          .    ACP "native" (unencrypted)
      + ACP auto-negotiated          .    "ACP connect subnet"
        and encrypted                .
                               ACP connect interface
                               e.g., "VRF ACP native" (config)
```
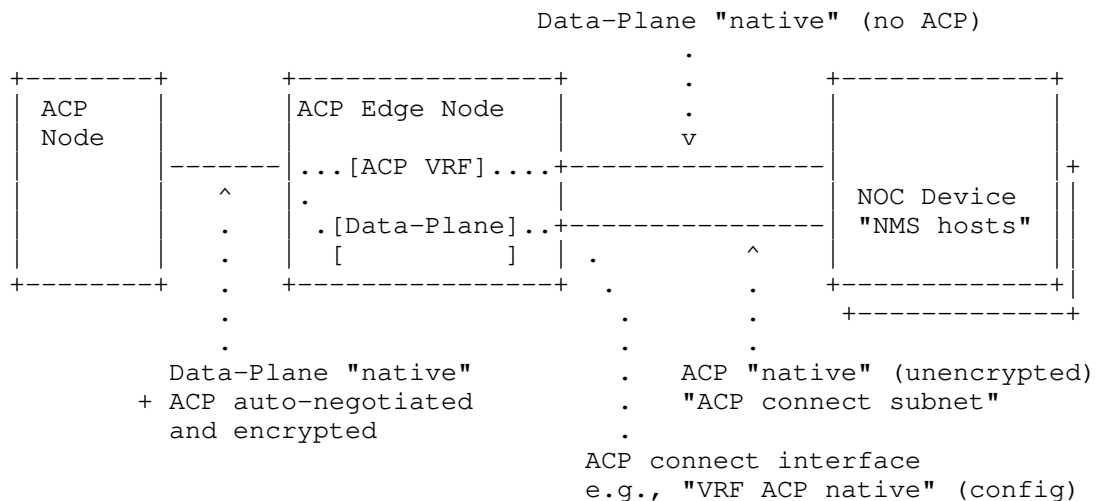
                       Figure 16: ACP connect

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

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

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

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

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

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

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

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

8.1.2. Software Components

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

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

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

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

8.1.3. Auto Configuration

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

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

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

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

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

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

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

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

                    Combined ACP and Data-Plane interface
                                        .
    +--------+   +-------------------+   .   +-------------+
    | ACP | |ACP Edge No | . | NMS Host(s) |
    | Node | | | . | / Software |
    | | | [ACP]. | . | | +
    | | | .[VRF] .[VRF   ] | v | "ACP address"| |
    | +--------+.   .[Select].+--------+ "Date Plane | |
    | | ^ | .[Data]. | | | Address(es)"| |
    | | . | [Plane] | | | | |
    | | . | [   ] | | +-------------+|
    +--------+ .   +-------------------+ | +-------------+
             .
          Data-Plane "native" and + ACP auto-negotiated/encrypted
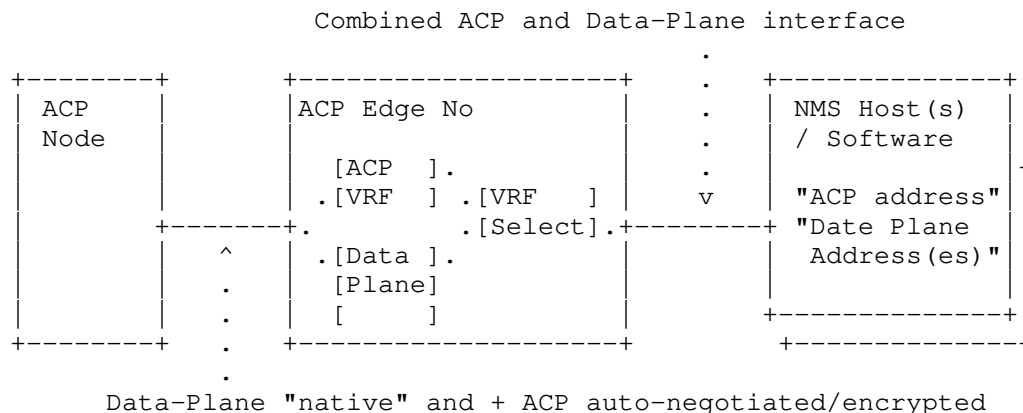
                          Figure 17: VRF select

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

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

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

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

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

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

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

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

8.1.5. Use of GRASP

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

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

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

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

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

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

8.2.1. Configured Remote ACP neighbor

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

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

                Figure 18: Parameters for remote ACP neighbors

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

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

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

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

8.2.2. Tunneled Remote ACP Neighbor

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

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

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

8.2.3. Summary

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

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

9. ACP Operations (Informative)

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

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

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

9.1. ACP (and BRSKI) Diagnostics

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

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

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

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

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

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

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

Q: Is ACP operational to accept neighbor connections:

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

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

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

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

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

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

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

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

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

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

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

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

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

9.1.1. Secure Channel Peer diagnostics

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

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

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

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

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

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

9.2. ACP Registrars

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

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

9.2.1. Registrar interactions

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

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

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

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

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

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

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

9.2.2. Registrar Parameter

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

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

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

9.2.3. Certificate renewal and limitations

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

This simple scenario has two limitations:

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

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

9.2.4. ACP Registrars with sub-CA

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

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

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

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

9.2.5. Centralized Policy Control

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

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

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

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

9.3. Enabling and disabling ACP/ANI

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

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

9.3.1. Filtering for non-ACP/ANI packets

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

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

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

9.3.2. Admin Down State

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

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

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

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

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

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

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

9.3.2.1. Security

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

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

9.3.2.2. Fast state propagation and Diagnostics

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

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

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

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

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

9.3.2.3. Low Level Link Diagnostics

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

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

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

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

9.3.2.4. Power Consumption Issues

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

9.3.3. Interface level ACP/ANI enable

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

9.3.4. Which interfaces to auto-enable?

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

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

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

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

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

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

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

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

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

9.3.5. Node Level ACP/ANI enable

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

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

9.3.5.1. Brownfield nodes

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

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

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

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

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

9.3.5.2. Greenfield nodes

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

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

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

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

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

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

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

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

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

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

9.3.6. Undoing ANI/ACP enable

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

9.3.7. Summary

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

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

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

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

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

9.4. Partial or Incremental adoption

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

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

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

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

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

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

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

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

9.5. Configuration and the ACP (summary)

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

The most important examples of such configuration include:

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

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

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

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

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

10. Summary: Benefits (Informative)

10.1. Self-Healing Properties

The ACP is self-healing:

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

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

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

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

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

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

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

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

10.2. Self-Protection Properties

10.2.1. From the outside

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

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

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

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

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

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

10.2.2. From the inside

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

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

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

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

10.3. The Administrator View

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

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

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

11. Security Considerations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

12. IANA Considerations

   This document defines the "Autonomic Control Plane".

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

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

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

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

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

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

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

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

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

13. Acknowledgements

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

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

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

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

14. Contributors

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

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

   Email: brian.e.carpenter@gmail.com

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

   Michael C. Richardson
   Sandelman Software Works

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

   For the RPL technology choices and text.

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

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

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

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

15.1. Summary of changes since entering IESG review

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

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

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

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

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

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

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

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

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

15.1.2. BRSKI / ACP registrar related enhancements

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

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

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

   6.1.4 (new) Overview of TA required for ACP.

   6.1.5.5 Added explanations/requirements for Re-enrollment.

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

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

15.1.3. Normative enhancements since start of IESG review

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

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

   6.1.5.3 Added explanations about CRLs.

   6.1.5.6 Added explanations of behavior under failing certificates.

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

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

6.6 Added throttling of secure-channel setup attempts.

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

6.12.5 Added recommendation to use IPv6 DAD.

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

15.1.4. Explanatory enhancements since start of IESG review

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

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

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

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

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

6.5 Elaborated step-by-step secure channel setup.

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

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

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

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

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

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

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

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

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

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

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

Comments from Erik Kline:

2. Fine tuned ULA definition.

Comments Michael Richardson / Eric Vyncke.

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

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

Comments/proposals from Ben Kaduk:

Various: Replaced RFC4492 with RFC8422 which is superceeding it.

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

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

6.2.5.4. Fine tuning 2 sentences.

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

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

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

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

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

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

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

A.6 - removed.

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

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

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

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

Other.

1. Added sentence about FCC outage report from June as example for
in-band management.

15. added reference to github where document was developed (removed
in RFC, part of changelog).

15.3. draft-ietf-anima-autonomic-control-plane-29

Comments from Robert Wilton:

Improved several textual nits.

Discuss/Comments from Erik Kline:

Editorial suggestions and nits. Thanks!.

6.1.3 Added text about how/why rsub is irrelevant for domain
membership check.

6.3 Added extension points to AN_ACP DULL GRASP objective because for
example ACP domain certificate could be a nice optional additional
parameter and prior syntax would have forced us to encode into
separate objective unnecessarily.

6.7 Using RFC8415 terminology for exponential backoff parameters.

6.11.2 Amended ACP Sub-Addressing table with future code points,
explanations and prefix announced into RPL.

6.12.1.11. Reworked text to better explain how black hole route
works and added expanation for prefix for manual address scheme.

8.1.3. Reworked explanation of RIOs for ACP connect interfaces for
Type C vs. Type A/B hosts.

8.1.4. Added explanation how this "VRF select" option is required
for auto-attachment of Type A/B hosts to ACP and other networks.

Discuss/Comments from Barry Leiba:

Various editorial nits - thanks.

6.1 New section pulling in TLS requirements, no need anymore to
duplicat for ACP GRASP, EST, BRSKI (ACP/ANI nodes) and (if desired)
OCSP/CRLDP. Added rule to start use secure channel only after
negotiation has finished. Added rules not to optimize negotiation
across multiple L2 interfaces to the same peer.

6.6 Changed role names in secure channel negotiation process: Alice/
Bob -> Decider/Follower. Explanation enhancements. Added definition
for ACP nodes with "0" address.

6.8.3 Improved explanation how IKEv2 forces preference of IPsec over
GRE due to ACP IPsec profiles being Tunneled vs. Transport.

6.8.4 Limited mentioning of DTLS version requirements to this
section.

6.9.2 Removed TLS requirements, they are now in 6.1.

6.10.6 Removed explanation of IANA allocation requirement. Redundant
- already in IANA section, and was seen as confusing.

8.1.1 Clarified that there can be security impacts when weakening
directly connected address RPF filtering for ACP connect interfaces.

Discuss/Comments from Ben Kaduk:

Many good editorial improvements - thanks!.

5. added explanation of what to do upon failed secure channel
establishment.

6.1.1. refined/extended cert public cey crypto algo and better
distinguished algo for the keys of the cert and the key of the
signer.

6.1.1. and following: explicitly defining "serialNumber" to be the
X.520 subject name serialNumber, not the certificate serial Number.

6.1.1. emhasize additional authorization step for EST servers (id-kp-
cmcRA).

6.1.2 changed AcpNodeName ABNF to again use 32HEXDIG instead of self-
defined variation, because authors overlooked that ABNF is case
agnostic (which is fine). Added recommendation to encode as lower
case. Added full ABNF encoding for extensions (any characters as in
"atoms" except the new "+" separator).

6.1.5.3. New text to explain reason for use of HTTPS (instead of
HTTP) for CRLDP and when and how to use HTTPS then.

6.1.5.5. added text explaning why/how and when to maintain TA data
upon failing cert renewal (one version with BRSKI, one version with
other, ess secure bootstrap protocols).

6.3. new text and requirement about the signaling of transport ports in DULL GRASP - benefits (no well-known ports required), and problems (additional DoS attack vector, albeit not worse than pre-existing ones, depending on setup of L2 subnets.).

6.7.3.1.1. Specified AUTH_HMAC_SHA2_256_128 (as the ESP authentication algorithm).

6.8.2. Added recommendations for TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256 when supporting TLS 1.3.

8.2.2. Added explanation about downgrade attack across configured ACP tunnels and what to do against it.

9.3.5.2. Rewrote most of section as it originally was too centric on BRSKI. Should now well describe expectations against automated bootstrap. Introduces new requirement not to call node as in support of ANI if is ALSO has TOFU bootstrap.

11. Expanded text about malicious EST servers. Added paragraph about ACP secure channel downgrade attacks. Added paragraphs about private PKI as a core to allow security against fake certificates, added paragraph about considerationsproblems when using public PI.

A.10.9 New appendix suggesting how to discover ACP secure channel negotiation downgrade attacks.

Discuss from Roman Danyliw:

6.1.5.1 - Added requirement to only announce SRV.est when a working CA connection.

15 - Amended security considerations with text about registrar dependencies, security of IDevID/ACP-certificate, EST-Server and GRASP for EST server discovery.

Other:

Conversion to XML v3. Solved empty () taxonomy xref problems. Various formatting fixes for v3.

Added contributors section.

15.4. draft-ietf-anima-autonomic-control-plane-28

IESG review Roman Danyliw:

6.   Requested additional text elaborating misconfiguration plus attack vectors.

6.1.3.1 Added paragraph about unsecured NTP as basis for time in the absence of other options.

6.7.2 reworded text about additional secure channel protocol requirements.

6.7.3.1.2. Added requirement for ACP nodes supporting IKEv2 to support RFC8247 (not sure how that got dropped from prior versions.

Replaced minimum crypto requirements definition via specific AES options with more generic "symmetric key/hash strength" requirements.

6.10.7.3. Added example how to derive addressing scheme from IDevID (PID). Added explanation how to deal with non-persistant registrar address database (hint: it sucks or is wasteful, what did you expect).

8.1.1. Added explanation for 'Physical controlled/secured'.

8.1.5. Removed 'Physical controlled/secured' text, refer back to 8.1.1.

8.2.1. Fixed ABNF 'or' syntax line.

9.3.2. Added explanation of remote management problem with interface "down" type commands.

10.2.1. Added explanations for attacks from impaired ACP nodes.

11. Rewrote intro paragraph. Removed text referring to enrollment/ registrars as they are out of scope of ACP (dependencies only).

11. Added note about need for new protocols inside ACP to use end- to-end authentication.

11. Rewrote paragraph about operator mistakes so as to be actionably. Operators must not make mistakes - but ACP minimizes the mistakes they can make.

ACP domain certificate -> ACP certificate.

Various other cosmetic edits (thanks!) and typo fixes (sorry for not running full spell check for every version. Will definitely do before RFC editor).

   Other:

   6.12.5.2.1./6.12.5.2.2. Added text explaining link breakage wrt.
   RTL (came about re-analyzing behavior after question about hop
   count).

   Removed now unnecessary references for earlier rrc822Name otherName
   choice.

15.5. draft-ietf-anima-autonomic-control-plane-27

   Too many revisions with too many fixes. Lets do a one-word change
   revision for a change now if it helps to accelerate the review
   process.

   Added "subjectAltName /" to make it unambiguous that AcpNodeName is
   indeed a SAN (from Russ).

15.6. draft-ietf-anima-autonomic-control-plane-26

   Russ Housley review of -25.

   1.1 Explicit reference for TLS 1.2 RFC.

   2. Changed term of "ACP Domain Information" to AcpNodeName (ASN.1) /
   acp-node-name (ABNF), also through rest of document.

   2. Improved CA behavior definition. changed IDevID/LDevID to IDevID/
   LDevID certificate to be more unambiguous.

   2. Changed definition of root CA to just refer to how its used in
   RFC7030 CA root key update, because thats the only thing relevant to
   ACP.

   6.1.1 Moved ECDH requirement to end of text as it was not related to
   the subject of the initial paragraps. Likewise reference to
   CABFORUM.

   6.1.1 Reduced cert key requirements to only be MUST for certs with
   2048 RSA public key and P-256 curves. Reduced longer keys to SHOULD.

   6.1.2 Changed text for conversion from rfc822Name to otherName /
   AcpNode, removed all the explanations of benefits coming with
   rfc822Name *sob* *sob* *sob*.

   6.1.2.1 New ASN.1 definition for otherName / AcpNodeName.

6.1.3 Fixed up text. re the handling of missing connectivity for CRLDP / OCSP.

6.1.4 Fixed up text re. inability to use public CA to situation with otherName / AcpNodeName (no more ACME rfc822Name validation for us *sob* *sob* *sob*).

12.   Added ASN.1 registration requests to IANA section.

Appenices. Minor changes for rfc822Name to otherName change.

Various minor verbal fixes/enhancements.

15.7. draft-ietf-anima-autonomic-control-plane-25

Crypto parameter discuss from Valery Smyslov and Paul Wouters and resulting changes.

6.7.2 Moved Michael Richardson suggested diagnostic of signaling TA from IPsec section to this general requirements section and added explanation how this may be inappropriate if TA payload is considered secret by TA owner.

6.7.3.1 Added traffic selectors for native IPsec. Improved text explanation.

6.7.3.1.2 removed misleading text about signaling TA when using intermediate certs.

6.7.3.1.2 Removed requirement for 'PKCS #7 wrapped X.509 certificate' requirement on request of Valery Smyslov as it is not defined in RFC7296 and there are enough options mandated in RFC7296. Replaced with just informative text to educate readers who are not IPsec experts what the mandatory option in RFC7296 is that allows to signal certificates.

6.7.3.1.2 Added SHOULD requirement how to deal with CERTREQ so that 6.7.2 requirement for TA diagnostics will work in IKEv2 (ignoring CERTREQ is permitted by IKEv2). Added explanation how this will result in TA cert diagnostics.

6.7.3.1.2 Added requirement for IKEv2 to operate on link-local addresses for ACP so at to assume ACT cert as the only possible authenticator – to avoid potentialy failing section from multiple available certs on a router.

6.7.3.1.2 fixed PKIX- style OID to ASN.1 object AlgorithmIdentifier (Paul).

6.7.3.2 Added IPsec traffic selectors for IPsec with GRE.

6.7.5 Added notion that IPsec/GRE MAY be preferred over IPsec/native. Luckily IPsec/native uses tunneling, whereas IPsec/GRE uses transport mode, and there is a long discuss whether it is permitted to even build IPsec connectings that only support transports instead of always being able to fall back to tunnel mode. Added explanatory paragraph why ACP nodes may prefer GRE over native (wonder how that was missing..).

9.1.1 Added section to explain need for secure channel peer diagnostics via signaling of TA. Four examples given.

Paul Wouters mentioned that ipkcs7 had to be used in some interop cases with windows CA, but that is an issue of ACP Registrar having to convert into PKCS#7 to talk to a windows CA, and this spec is not concerned with that, except to know that it is feasible, so not mentioned in text anywhere, just tracking discussion here in changelog.

Michael Richardson:

3.1.3 Added point in support of rfc822address that CA may not support to sign certificates with new attributes (such as new otherName).

Michael Richardson/Brian Carpenter fix:

6.1.5.1/6.3 Fixed GRASP examples.

Joe Halpern review:

1. Enhanded introduction text for in-band and of out-of-band, explaining how ACP is an in-band network aiming to achieve all possible benefits of an out-of-band network.

1. Comprehensive explanation for term Data-Plane as it is only logically following pre-established terminology on a fully autonomic node, when used for existing nodes augmented with ACP, Data-Plane has more functionality than usually associated with the term.

2. Removed explanatory text for Data-Plane, referring to section 1.

2. Reduced explanation in definition of in-band (management/ signaling), out-of-band-signaling, now pointing to section 1.

5. Rewrote a lot of the steps (overview) as this text was not reviewed for long time. Added references to normative section for each step to hopefully avoid feedback of not explaining terms used (really not possible to give good summary without using forward references).

2. Separate out-of-band-management definition from virtual out-of-band-management definition (later one for ACP).

2. Added definitions for RPI and RPL.

6.1.1. added note about end-to-end authentication to distinguish channel security from overall ACP security model.

6.5 Fixed bugs in channel selection signaling step description (Alice vs. Bob).

6.7.1 Removed redundant channel selection explanation.

6.10.3 remove locator/identifier terminology from zone addressing scheme description (unnecessary), removed explanations (now in 9.4), simplified text, clarified requirement for Node-ID to be unique, recommend to use primarily zone 0.

6.10.3.1 Removed. Included a lot of insufficient suggestions for future stanards extensions, most of it was wrong or would need to be revisited by WG anyhow. Idea now (just here for comment): Announce via GRASP Zone-ID (e.g. from per-zone edge-node/registrar) into a zone of the ACP so all nodes supporting the scheme can automatically self-allocate the Zone-ID.

6.11.1.1 (RPL overview), eliminated redundant text.

6.11.1.1.1 New subsection to better structure overview.

6.11.1.1.2 New subsection to better group overview, replaced TTL explanation (just the symptom) with hopefully better reconvergence text (intent of the profile) for the ACP RPL profile.

6.11.1.1.6 Added text to explain simple choice for rank_factor.

6.11.1.13 moved explanation for RPI up into 6.11.1.1.

6.12.5.1 rewrote section for ACP Loopback Interface.

9.4 New informative/informational section for partial or incremental adoption of ACP to help understand why there is the Zone interface sub-scheme, and how to use it.

Unrelated fixes:

Ask to RFC editor to add most important abbreviations to RFC editor abbreviation list.

6.10.2 changed names in ACP addressing scheme table to be less suggestive of use.

Russ Hously review:

2. Fixed definition of "Enrollment", "Trust Anchor", "CA", and "root CA". Changed "Certificate Authority" to "Certification Authority" throughout the document (correct term according to X.509).

6.1 Fixed explanation of mutual ACP trust.

6.1.1 s/X509/X509v3/.

6.1.2 created bulleted lists for explanations and justifications for choices of ACP certificate encoding. No semantic changes, just to make it easier to refer to the points in discussions (rfcdiff seems to have a bug showing text differences due to formatting changes).

6.1.3 Moved content of rule #1 into previous rule #2 because certification chain validation does imply validation of lifetime. numbers of all rules reduced by 1, changed hopefully all references to the rule numbers in the document.

Rule #3, Hopefully fixed linguistic problem self-contradiction of MUST by lower casing MUST in the explanation part and rewriting the condition when this is not applicable.

6.1.4 Replaced redundant term "Trust Point" (TP) with Trust Anchor (TA"). Replaced throughout document Trust Anchor with abbreviation TA.

Enhanced several sentences/rewrote paragraphs to make explanations clearer.

6.6 Added explanation how ACP nodes must throttle their attempts for connection making purely on the result of their own connection attempts, not based on those connections where they are responder.

15.8. draft-ietf-anima-autonomic-control-plane-24

Leftover from -23 review by Eric Vyncke:

Swapping sections 9 and 10, section 9 was meant to be at end of document and summarize. Its not meant to be misinterpreted as introducing any new information. This did happen because section 10 was added after section 9.

15.9. draft-ietf-anima-autonomic-control-plane-23

Note: big rfcdiff of TOC is an rfcdiff bug, changes really minimal.

Review of IPsec security with Mcr and ipsec mailing list.

6.7.1 - new section: Moved general considerations for secure channel protocols here, refined them.

6.7.2 - new section: Moved common requirements for secure channel protocols here, refined them.

6.7.3.1.1. - improved requirements text related to RFC8221, better explamations re. HW acceleration issues.

6.7.3.1.2. - improved requirements text related to RFC8247, (some requirements still discussed to be redundant, will be finalized in next weeks.

Eric Vyncke review of -21:

Only noting most important changes, long list of smaller text/ readability enhancements.

2. - New explanation of "normative", "informational" section title tags. alphabetic reordering of terms, refined definitions for CA, CRL. root CA.

6.1.1. - explanation when IDevID parameters may be copied into LDevID.

6.1.2. - Fixed hex digits in ACP domain information to lower case.

6.1.3.1. - New section on Realtime clock and Time Validation.

6.3 - Added explanation that DTLS means >= version 1.2 (not only 1.2).

6.7 - New text in this main section explaing relationship of ACP secure channels and ACP virtual interfaces - with forward references to virtual interface section.

6.8.2 - reordered text and picture, no text change.

6.10.7.2 - describe first how Registrar-ID can be allocted for all type of registrars, then refined text for how to potentially use MAC addresses on physical registrars.

6.11.1.1 - Added text how this profile does not use Data-Plane artefacts (RPI) because hadware forwarding. This was previously hidden only later in the text.

6.11.1.13. - Rewrote RPL Data-Plane artefact text. Provide decoder ring for abbreviations and all relevant RFCs.

6.12.5.2. - Added more explicit text that secure channels are mapped into virtual interfaces, moved different type of interfaces used by ACP into separate subsections to be able to refer to them.

7.2 - Rewrote/refined text for ACP on L2, prior text was confusing and did not well explain why ACP for L2/L3 switches can be implemented without any L2 (HW) changes. Also missing explanation of only running GRASP untagged when VLANs are used.

8.1.1 - Added requirement for ACP Edge nodes to allow configurable filtering of IPv6 RPI headers.

11. - (security section). Moved explanation of address stealing from 7.2 to here.

15.10. draft-ietf-anima-autonomic-control-plane-22

Ben Kaduk review of -21:

RFC822 encoding of ACP domain information:

6.1.2 rewrote text for explaining / justifying use of rfc822name as identifier for node CP in certificate (was discussed in thread, but badly written in prior versions).

6.1.2 Changed EBNF syntax to use "+" after rfcSELF because that is the known primary name to extensions separator in many email systems ("." was wrong in prior versions).

6.1.2 Rewrote/improved explanations for use of rfc822name field to explain better why it is PKIX compliant and the right thing to do.

Crypto parameters for IPsec:

6.1 - Added explanation of why manual keying for ACP is not feasible
for ACP. Surprisingly, that text did not exist. Referred to by
IPsec text (6.7.1), but here is the right place to describe the
reasoning.

6.1.2 - Small textual refinement referring to requirements to
authenticate peers (for the special cases of empty or '0' ACP address
in ACP domain information field.

6.3 - To better justify Bens proposed change of secure channel
protocol being IPsec vs. GRASP objective being IKEv2, better
explained how protocol indicated in GRASP objective-value is name of
protocol used to negotiate secure channel, use example of IKEv2 to
negotiate IPsec.

6.7.1 - refinemenet similar to 6.3.

- moved new paragraph from Bens pull request up from 6.7.1.1 to 6.7.1
as it equally applies to GRE encapped IPsec (looks nicer one level
up).

- created subsections 6.7.1.1 (IPsec/ESP) / 6.7.1.2 (IKEv2) to
clearer distinguish between these two requirements blocks.

- Refined the text in these two sections to hopefully be a good
answer to Valery's concern of not randomnly mocking with existing
requirements docs (rfc8247 / rfc8221).

6.7.1.1.1 - IPsec/ESP requirements section:

- MUST support rfc8221 mandatory EXCEPT for the superceeding
requirements in this section. Previously, this was not quite clear
from the text.

- Hopefully persuasive explanations about the requirements levels for
ENCR_AES_GCM_16, ENCR_AES_CBC, ENCR_AES_CCM_8 and
ENCR_CHACHA20_POLY1305: Restructured text for why not ENCR_AES_CBC
(was in prior version, just not well structured), added new
expanations for ENCR_AES_CCM_8 and ENCR_CHACHA20_POLY130.

- In simple terms, requirements for ENCR_AES_CBC, ENCR_AES_CCM_8,
ENCR_CHACHACHA are SHOULD when they are implementable with rqual or
faster performancce than ENCR_AES_GCM_16.

- Removed text about "additional rfc8221" reqiurements MAY be used.
Now the logic is that all other requirements apply. Hopefully we
have written enough so that we prohibited downgrades.

6.7.1.1.2 - RFC8247 requirements:

- Added mandate to support rfc8247, added explanation that there is
no "stripping down" requirement, just additional stronger
requirements to mandate correct use of ACP certificartes during
authentication.

- refined text on identifying ACP by IPv6 address to be clearer:
Identifying in the context of IKEv2 and cases for '0' in ACP domain
information.

- removed last two paragraphs about relationship to rfc8247, as his
is now written in first paragraph of the section.

End of Ben Kaduk review related fixes.

Other:

Forgot to update example of ACP domain information to use capitalized
hex-digits as required by HEXDIG used.

Added reference to RFC8316 (AN use-cases) to beginning of section 3
(ACP use cases).

Small Enhanced IPsec parameters description / requirements fixes
(from Michael Richardson).

16. Normative References

[I-D.ietf-anima-bootstrapping-keyinfra]
           Pritikin, M., Richardson, M., Eckert, T., Behringer, M.,
           and K. Watsen, "Bootstrapping Remote Secure Key
           Infrastructures (BRSKI)", Work in Progress, Internet-
           Draft, draft-ietf-anima-bootstrapping-keyinfra-43, 7
           August 2020, <http://www.ietf.org/internet-drafts/draft-
           ietf-anima-bootstrapping-keyinfra-43.txt>.

[I-D.ietf-anima-grasp]
           Bormann, C., Carpenter, B., and B. Liu, "A Generic
           Autonomic Signaling Protocol (GRASP)", Work in Progress,
           Internet-Draft, draft-ietf-anima-grasp-15, 13 July 2017,
           <http://www.ietf.org/internet-drafts/draft-ietf-anima-
           grasp-15.txt>.

[IKEV2IANA]
           IANA, "Internet Key Exchange Version 2 (IKEv2)
           Parameters", <https://www.iana.org/assignments/ikev2-
           parameters/ikev2-parameters.xhtml>.

   [RFC1034] Mockapetris, P.V., "Domain names - concepts and
              facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034,
              November 1987, <https://www.rfc-editor.org/info/rfc1034>.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <https://www.rfc-editor.org/info/rfc2119>.

   [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener
              Discovery Version 2 (MLDv2) for IPv6", RFC 3810,
              DOI 10.17487/RFC3810, June 2004,
              <https://www.rfc-editor.org/info/rfc3810>.

   [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and
              More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191,
              November 2005, <https://www.rfc-editor.org/info/rfc4191>.

   [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast
              Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005,
              <https://www.rfc-editor.org/info/rfc4193>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, DOI 10.17487/RFC4291, February
              2006, <https://www.rfc-editor.org/info/rfc4291>.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, DOI 10.17487/RFC4301,
              December 2005, <https://www.rfc-editor.org/info/rfc4301>.

   [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman,
              "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
              DOI 10.17487/RFC4861, September 2007,
              <https://www.rfc-editor.org/info/rfc4861>.

   [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless
              Address Autoconfiguration", RFC 4862,
              DOI 10.17487/RFC4862, September 2007,
              <https://www.rfc-editor.org/info/rfc4862>.

   [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", STD 68, RFC 5234,
              DOI 10.17487/RFC5234, January 2008,
              <https://www.rfc-editor.org/info/rfc5234>.

[RFC5246]   Dierks, T. and E. Rescorla, "The Transport Layer Security
            (TLS) Protocol Version 1.2", RFC 5246,
            DOI 10.17487/RFC5246, August 2008,
            <https://www.rfc-editor.org/info/rfc5246>.

[RFC5280]   Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
            Housley, R., and W. Polk, "Internet X.509 Public Key
            Infrastructure Certificate and Certificate Revocation List
            (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
            <https://www.rfc-editor.org/info/rfc5280>.

[RFC6347]   Rescorla, E. and N. Modadugu, "Datagram Transport Layer
            Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
            January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6550]   Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J.,
            Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur,
            JP., and R. Alexander, "RPL: IPv6 Routing Protocol for
            Low-Power and Lossy Networks", RFC 6550,
            DOI 10.17487/RFC6550, March 2012,
            <https://www.rfc-editor.org/info/rfc6550>.

[RFC6551]   Vasseur, JP., Ed., Kim, M., Ed., Pister, K., Dejean, N.,
            and D. Barthel, "Routing Metrics Used for Path Calculation
            in Low-Power and Lossy Networks", RFC 6551,
            DOI 10.17487/RFC6551, March 2012,
            <https://www.rfc-editor.org/info/rfc6551>.

[RFC6552]   Thubert, P., Ed., "Objective Function Zero for the Routing
            Protocol for Low-Power and Lossy Networks (RPL)",
            RFC 6552, DOI 10.17487/RFC6552, March 2012,
            <https://www.rfc-editor.org/info/rfc6552>.

[RFC6553]   Hui, J. and JP. Vasseur, "The Routing Protocol for Low-
            Power and Lossy Networks (RPL) Option for Carrying RPL
            Information in Data-Plane Datagrams", RFC 6553,
            DOI 10.17487/RFC6553, March 2012,
            <https://www.rfc-editor.org/info/rfc6553>.

[RFC7030]   Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
            "Enrollment over Secure Transport", RFC 7030,
            DOI 10.17487/RFC7030, October 2013,
            <https://www.rfc-editor.org/info/rfc7030>.

[RFC7296]   Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T.
            Kivinen, "Internet Key Exchange Protocol Version 2
            (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October
            2014, <https://www.rfc-editor.org/info/rfc7296>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

   [RFC7676] Pignataro, C., Bonica, R., and S. Krishnan, "IPv6 Support
              for Generic Routing Encapsulation (GRE)", RFC 7676,
              DOI 10.17487/RFC7676, October 2015,
              <https://www.rfc-editor.org/info/rfc7676>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T.
              Kivinen, "Cryptographic Algorithm Implementation
              Requirements and Usage Guidance for Encapsulating Security
              Payload (ESP) and Authentication Header (AH)", RFC 8221,
              DOI 10.17487/RFC8221, October 2017,
              <https://www.rfc-editor.org/info/rfc8221>.

   [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault,
              "Algorithm Implementation Requirements and Usage Guidance
              for the Internet Key Exchange Protocol Version 2 (IKEv2)",
              RFC 8247, DOI 10.17487/RFC8247, September 2017,
              <https://www.rfc-editor.org/info/rfc8247>.

   [RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic
              Curve Cryptography (ECC) Cipher Suites for Transport Layer
              Security (TLS) Versions 1.2 and Earlier", RFC 8422,
              DOI 10.17487/RFC8422, August 2018,
              <https://www.rfc-editor.org/info/rfc8422>.

   [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
              <https://www.rfc-editor.org/info/rfc8446>.

   [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data
              Definition Language (CDDL): A Notational Convention to
              Express Concise Binary Object Representation (CBOR) and
              JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610,
              June 2019, <https://www.rfc-editor.org/info/rfc8610>.

17. Informative References

   [ACPDRAFT] Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic
               Control Plane (ACP)", Work in Progress, Internet-Draft,
               draft-ietf-anima-autonomic-control-plane-30,
               <https://tools.ietf.org/html/draft-ietf-anima-autonomic-
               control-plane-30.pdf>. [RFC-Editor: Please remove this
               complete reference from the RFC] Refer to the IETF working
               group draft for the few sections removed from this
               document for various reasons. They capture the state of
               discussion about unresolved issues that may need to be
               revisited in future work.

   [AR8021]   Group, W. -. H. L. L. P. W., "IEEE Standard for Local and
               metropolitan area networks - Secure Device Identity",
               December 2009, <http://standards.ieee.org/findstds/
               standard/802.1AR-2009.html>.

   [CABFORUM] CA/Browser Forum, "Certificate Contents for Baseline SSL",
               November 2019, <https://cabforum.org/baseline-
               requirements-certificate-contents/>.

   [FCC]   FCC, "FCC STAFF REPORT ON NATIONWIDE T-MOBILE NETWORK
               OUTAGE ON JUNE 15, 2020 (PS Docket No. 20-183)", 2020,
               <https://docs.fcc.gov/public/attachments/DOC-
               367699A1.docx>. The FCC's Public Safety and Homeland
               Security Bureau issues a report on a nationwide T-Mobile
               outage that occurred on June 15, 2020. Action by: Public
               Safety and Homeland Security Bureau.

   [I-D.eckert-anima-noc-autoconfig]
               Eckert, T., "Autoconfiguration of NOC services in ACP
               networks via GRASP", Work in Progress, Internet-Draft,
               draft-eckert-anima-noc-autoconfig-00, 2 July 2018,
               <http://www.ietf.org/internet-drafts/draft-eckert-anima-
               noc-autoconfig-00.txt>.

   [I-D.ietf-acme-star]
               Sheffer, Y., Lopez, D., Dios, O., Pastor, A., and T.
               Fossati, "Support for Short-Term, Automatically-Renewed
               (STAR) Certificates in Automated Certificate Management
               Environment (ACME)", Work in Progress, Internet-Draft,
               draft-ietf-acme-star-11, 24 October 2019,
               <http://www.ietf.org/internet-drafts/draft-ietf-acme-star-
               11.txt>.

   [I-D.ietf-anima-prefix-management]
               Jiang, S., Du, Z., Carpenter, B., and Q. Sun, "Autonomic
               IPv6 Edge Prefix Management in Large-scale Networks", Work
               in Progress, Internet-Draft, draft-ietf-anima-prefix-

                  management-07, 18 December 2017, <http://www.ietf.org/
                  internet-drafts/draft-ietf-anima-prefix-management-
                  07.txt>.

     [I-D.ietf-anima-reference-model]
                  Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L.,
                  and J. Nobre, "A Reference Model for Autonomic
                  Networking", Work in Progress, Internet-Draft, draft-ietf-
                  anima-reference-model-10, 22 November 2018,
                  <http://www.ietf.org/internet-drafts/draft-ietf-anima-
                  reference-model-10.txt>.

     [I-D.ietf-roll-applicability-template]
                  Richardson, M., "ROLL Applicability Statement Template",
                  Work in Progress, Internet-Draft, draft-ietf-roll-
                  applicability-template-09, 3 May 2016,
                  <http://www.ietf.org/internet-drafts/draft-ietf-roll-
                  applicability-template-09.txt>.

     [I-D.ietf-tls-dtls13]
                  Rescorla, E., Tschofenig, H., and N. Modadugu, "The
                  Datagram Transport Layer Security (DTLS) Protocol Version
                  1.3", Work in Progress, Internet-Draft, draft-ietf-tls-
                  dtls13-38, 29 May 2020, <http://www.ietf.org/internet-
                  drafts/draft-ietf-tls-dtls13-38.txt>.

     [IEEE-1588-2008]
                  IEEE, "IEEE Standard for a Precision Clock Synchronization
                  Protocol for Networked Measurement and Control Systems",
                  December 2008, <http://standards.ieee.org/findstds/
                  standard/1588-2008.html>.

     [IEEE-802.1X]
                  Group, W. -. H. L. L. P. W., "IEEE Standard for Local and
                  Metropolitan Area Networks: Port-Based Network Access
                  Control", February 2010,
                  <http://standards.ieee.org/findstds/standard/802.1X-
                  2010.html>.

     [LLDP]   Group, W. -. H. L. L. P. W., "IEEE Standard for Local and
                  Metropolitan Area Networks: Station and Media Access
                  Control Connectivity Discovery", June 2016,
                  <https://standards.ieee.org/findstds/standard/802.1AB-
                  2016.html>.

   [MACSEC]   Group, W. -. H. L. L. P. W., "IEEE Standard for Local and
              Metropolitan Area Networks: Media Access Control (MAC)
              Security", June 2006,
              <https://standards.ieee.org/findstds/standard/802.1AE-
              2006.html>.

   [RFC1112] Deering, S.E., "Host extensions for IP multicasting",
              STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989,
              <https://www.rfc-editor.org/info/rfc1112>.

   [RFC1492] Finseth, C., "An Access Control Protocol, Sometimes Called
              TACACS", RFC 1492, DOI 10.17487/RFC1492, July 1993,
              <https://www.rfc-editor.org/info/rfc1492>.

   [RFC1654] Rekhter, Y., Ed. and T. Li, Ed., "A Border Gateway
              Protocol 4 (BGP-4)", RFC 1654, DOI 10.17487/RFC1654, July
              1994, <https://www.rfc-editor.org/info/rfc1654>.

   [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.
              J., and E. Lear, "Address Allocation for Private
              Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918,
              February 1996, <https://www.rfc-editor.org/info/rfc1918>.

   [RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax
              Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998,
              <https://www.rfc-editor.org/info/rfc2315>.

   [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange
              (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998,
              <https://www.rfc-editor.org/info/rfc2409>.

   [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson,
              "Remote Authentication Dial In User Service (RADIUS)",
              RFC 2865, DOI 10.17487/RFC2865, June 2000,
              <https://www.rfc-editor.org/info/rfc2865>.

   [RFC3164] Lonvick, C., "The BSD Syslog Protocol", RFC 3164,
              DOI 10.17487/RFC3164, August 2001,
              <https://www.rfc-editor.org/info/rfc3164>.

   [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
              C., and M. Carney, "Dynamic Host Configuration Protocol
              for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
              2003, <https://www.rfc-editor.org/info/rfc3315>.

   [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An
              Architecture for Describing Simple Network Management
              Protocol (SNMP) Management Frameworks", STD 62, RFC 3411,
              DOI 10.17487/RFC3411, December 2002,
              <https://www.rfc-editor.org/info/rfc3411>.

   [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi,
              "DNS Extensions to Support IP Version 6", STD 88,
              RFC 3596, DOI 10.17487/RFC3596, October 2003,
              <https://www.rfc-editor.org/info/rfc3596>.

   [RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence
              Protocol (XMPP): Core", RFC 3920, DOI 10.17487/RFC3920,
              October 2004, <https://www.rfc-editor.org/info/rfc3920>.

   [RFC3954] Claise, B., Ed., "Cisco Systems NetFlow Services Export
              Version 9", RFC 3954, DOI 10.17487/RFC3954, October 2004,
              <https://www.rfc-editor.org/info/rfc3954>.

   [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and
              B. Zill, "IPv6 Scoped Address Architecture", RFC 4007,
              DOI 10.17487/RFC4007, March 2005,
              <https://www.rfc-editor.org/info/rfc4007>.

   [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen,
              "Internet X.509 Public Key Infrastructure Certificate
              Management Protocol (CMP)", RFC 4210,
              DOI 10.17487/RFC4210, September 2005,
              <https://www.rfc-editor.org/info/rfc4210>.

   [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private
              Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February
              2006, <https://www.rfc-editor.org/info/rfc4364>.

   [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD)
              for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006,
              <https://www.rfc-editor.org/info/rfc4429>.

   [RFC4541] Christensen, M., Kimball, K., and F. Solensky,
              "Considerations for Internet Group Management Protocol
              (IGMP) and Multicast Listener Discovery (MLD) Snooping
              Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006,
              <https://www.rfc-editor.org/info/rfc4541>.

   [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet
              Group Management Protocol Version 3 (IGMPv3) and Multicast
              Listener Discovery Protocol Version 2 (MLDv2) for Source-
              Specific Multicast", RFC 4604, DOI 10.17487/RFC4604,
              August 2006, <https://www.rfc-editor.org/info/rfc4604>.

   [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for
              IP", RFC 4607, DOI 10.17487/RFC4607, August 2006,
              <https://www.rfc-editor.org/info/rfc4607>.

   [RFC4610] Farinacci, D. and Y. Cai, "Anycast-RP Using Protocol
              Independent Multicast (PIM)", RFC 4610,
              DOI 10.17487/RFC4610, August 2006,
              <https://www.rfc-editor.org/info/rfc4610>.

   [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy
              Extensions for Stateless Address Autoconfiguration in
              IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007,
              <https://www.rfc-editor.org/info/rfc4941>.

   [RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure
              Subject Alternative Name for Expression of Service Name",
              RFC 4985, DOI 10.17487/RFC4985, August 2007,
              <https://www.rfc-editor.org/info/rfc4985>.

   [RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet
              Group Management Protocol Version 3 (IGMPv3) and Multicast
              Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790,
              DOI 10.17487/RFC5790, February 2010,
              <https://www.rfc-editor.org/info/rfc5790>.

   [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
              <https://www.rfc-editor.org/info/rfc5880>.

   [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch,
              "Network Time Protocol Version 4: Protocol and Algorithms
              Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010,
              <https://www.rfc-editor.org/info/rfc5905>.

   [RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the
              Public Key Infrastructure Using X.509 (PKIX)", RFC 5912,
              DOI 10.17487/RFC5912, June 2010,
              <https://www.rfc-editor.org/info/rfc5912>.

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
              and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <https://www.rfc-editor.org/info/rfc6241>.

   [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S.
              Cheshire, "Internet Assigned Numbers Authority (IANA)
              Procedures for the Management of the Service Name and
              Transport Protocol Port Number Registry", BCP 165,
              RFC 6335, DOI 10.17487/RFC6335, August 2011,
              <https://www.rfc-editor.org/info/rfc6335>.

   [RFC6402] Schaad, J., "Certificate Management over CMS (CMC)
              Updates", RFC 6402, DOI 10.17487/RFC6402, November 2011,
              <https://www.rfc-editor.org/info/rfc6402>.

   [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain
              of Interpretation", RFC 6407, DOI 10.17487/RFC6407,
              October 2011, <https://www.rfc-editor.org/info/rfc6407>.

   [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6
              Routing Header for Source Routes with the Routing Protocol
              for Low-Power and Lossy Networks (RPL)", RFC 6554,
              DOI 10.17487/RFC6554, March 2012,
              <https://www.rfc-editor.org/info/rfc6554>.

   [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
              "Default Address Selection for Internet Protocol Version 6
              (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
              <https://www.rfc-editor.org/info/rfc6724>.

   [RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn,
              Ed., "Diameter Base Protocol", RFC 6733,
              DOI 10.17487/RFC6733, October 2012,
              <https://www.rfc-editor.org/info/rfc6733>.

   [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762,
              DOI 10.17487/RFC6762, February 2013,
              <https://www.rfc-editor.org/info/rfc6762>.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <https://www.rfc-editor.org/info/rfc6763>.

   [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure,
              "TCP Extensions for Multipath Operation with Multiple
              Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013,
              <https://www.rfc-editor.org/info/rfc6824>.

   [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The
              Locator/ID Separation Protocol (LISP)", RFC 6830,
              DOI 10.17487/RFC6830, January 2013,
              <https://www.rfc-editor.org/info/rfc6830>.

   [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken,
              "Specification of the IP Flow Information Export (IPFIX)
              Protocol for the Exchange of Flow Information", STD 77,
              RFC 7011, DOI 10.17487/RFC7011, September 2013,
              <https://www.rfc-editor.org/info/rfc7011>.

   [RFC7404] Behringer, M. and E. Vyncke, "Using Only Link-Local
              Addressing inside an IPv6 Network", RFC 7404,
              DOI 10.17487/RFC7404, November 2014,
              <https://www.rfc-editor.org/info/rfc7404>.

   [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S.,
              Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-
              Defined Networking (SDN): Layers and Architecture
              Terminology", RFC 7426, DOI 10.17487/RFC7426, January
              2015, <https://www.rfc-editor.org/info/rfc7426>.

   [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection
              Most of the Time", RFC 7435, DOI 10.17487/RFC7435,
              December 2014, <https://www.rfc-editor.org/info/rfc7435>.

   [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A.,
              Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic
              Networking: Definitions and Design Goals", RFC 7575,
              DOI 10.17487/RFC7575, June 2015,
              <https://www.rfc-editor.org/info/rfc7575>.

   [RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap
              Analysis for Autonomic Networking", RFC 7576,
              DOI 10.17487/RFC7576, June 2015,
              <https://www.rfc-editor.org/info/rfc7576>.

   [RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for
              RADIUS/TLS and RADIUS/DTLS Based on the Network Access
              Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October
              2015, <https://www.rfc-editor.org/info/rfc7585>.

   [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy
              Considerations for IPv6 Address Generation Mechanisms",
              RFC 7721, DOI 10.17487/RFC7721, March 2016,
              <https://www.rfc-editor.org/info/rfc7721>.

   [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I.,
              Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent
              Multicast - Sparse Mode (PIM-SM): Protocol Specification
              (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March
              2016, <https://www.rfc-editor.org/info/rfc7761>.

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

   [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by
              Hosts in a Multi-Prefix Network", RFC 8028,
              DOI 10.17487/RFC8028, November 2016,
              <https://www.rfc-editor.org/info/rfc8028>.

   [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
              Writing an IANA Considerations Section in RFCs", BCP 26,
              RFC 8126, DOI 10.17487/RFC8126, June 2017,
              <https://www.rfc-editor.org/info/rfc8126>.

   [RFC8316] Nobre, J., Granville, L., Clemm, A., and A. Gonzalez
              Prieto, "Autonomic Networking Use Case for Distributed
              Detection of Service Level Agreement (SLA) Violations",
              RFC 8316, DOI 10.17487/RFC8316, February 2018,
              <https://www.rfc-editor.org/info/rfc8316>.

   [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,
              "A Voucher Artifact for Bootstrapping Protocols",
              RFC 8366, DOI 10.17487/RFC8366, May 2018,
              <https://www.rfc-editor.org/info/rfc8366>.

   [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic
              Control Plane for Stable Connectivity of Network
              Operations, Administration, and Maintenance (OAM)",
              RFC 8368, DOI 10.17487/RFC8368, May 2018,
              <https://www.rfc-editor.org/info/rfc8368>.

   [RFC8398] Melnikov, A., Ed. and W. Chuang, Ed., "Internationalized
              Email Addresses in X.509 Certificates", RFC 8398,
              DOI 10.17487/RFC8398, May 2018,
              <https://www.rfc-editor.org/info/rfc8398>.

   [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,
              Decraene, B., Litkowski, S., and R. Shakir, "Segment
              Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
              July 2018, <https://www.rfc-editor.org/info/rfc8402>.

   [RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero
              Touch Provisioning (SZTP)", RFC 8572,
              DOI 10.17487/RFC8572, April 2019,
              <https://www.rfc-editor.org/info/rfc8572>.

   [X.509]   International Telecommunication Union, "Information
              technology - Open Systems Interconnection - The Directory:
              Public-key and attribute certificate frameworks", ITU-T
              Recommendation X.509, ISO/IEC 9594-8, October 2016,
              <https://www.itu.int/rec/T-REC-X.509>.

   [X.520]   International Telecommunication Union, "Information
              technology - Open Systems Interconnection - The Directory:
              Selected attribute types", ITU-T Recommendation
              X.520, ISO/IEC 9594-6, October 2016,
              <https://www.itu.int/rec/T-REC-X.520>.

Appendix A. Background and Futures (Informative)

   The following sections discuss additional background information
   about aspects of the normative parts of this document or associated
   mechanisms such as BRSKI (such as why specific choices were made by
   the ACP) and they provide discussion about possible future variations
   of the ACP.

A.1. ACP Address Space Schemes

   This document defines the Zone, Vlong and Manual sub address schemes
   primarily to support address prefix assignment via distributed,
   potentially uncoordinated ACP registrars as defined in
   Section 6.11.7. This costs 48/46-bit identifier so that these ACP
   registrar can assign non-conflicting address prefixes. This design
   does not leave enough bits to simultaneously support a large number
   of nodes (Node-ID) plus a large prefix of local addresses for every
   node plus a large enough set of bits to identify a routing Zone. In
   result, Zone, Vlong 8/16 attempt to support all features, but via
   separate prefixes.

In networks that always expect to rely on a centralized PMS as
described above (Section 9.2.5), the 48/46-bits for the Registrar-ID
could be saved. Such variations of the ACP addressing mechanisms
could be introduced through future work in different ways. If a new
otherName was introduced, incompatible ACP variations could be
created where every design aspect of the ACP could be changed.
Including all addressing choices. If instead a new addressing sub-
type would be defined, it could be a backward compatible extension of
this ACP specification. Information such as the size of a zone-
prefix and the length of the prefix assigned to the ACP node itself
could be encoded via the extension field of the acp-node-name.

Note that an explicitly defined "Manual" addressing sub-scheme is
always beneficial to provide an easy way for ACP nodes to prohibit
incorrect manual configuration of any non-"Manual" ACP address spaces
and therefore ensure that "Manual" operations will never impact
correct routing for any non-"Manual" ACP addresses assigned via ACP
certificates.

A.2. BRSKI Bootstrap (ANI)

BRSKI describes how nodes with an IDevID certificate can securely and
zero-touch enroll with an LDevID certificate to support the ACP.
BRSKI also leverages the ACP to enable zero-touch bootstrap of new
nodes across networks without any configuration requirements across
the transit nodes (e.g., no DHCP/DNS forwarding/server setup). This
includes otherwise not configured networks as described in
Section 3.2. Therefore, BRSKI in conjunction with ACP provides for a
secure and zero-touch management solution for complete networks.
Nodes supporting such an infrastructure (BRSKI and ACP) are called
ANI nodes (Autonomic Networking Infrastructure), see
[I-D.ietf-anima-reference-model]. Nodes that do not support an
IDevID certificate but only an (insecure) vendor specific Unique
Device Identifier (UDI) or nodes whose manufacturer does not support
a MASA could use some future security reduced version of BRSKI.

When BRSKI is used to provision a domain certificate (which is called
enrollment), the BRSKI registrar (acting as an enhanced EST server)
must include the otherName / AcpNodeName encoded ACP address and
domain name to the enrolling node (called pledge) via its response to
the pledges EST CSR Attribute request that is mandatory in BRSKI.

The Certification Authority in an ACP network must not change the
otherName / AcpNodeName in the certificate. The ACP nodes can
therefore find their ACP address and domain using this field in the
domain certificate, both for themselves, as well as for other nodes.

The use of BRSKI in conjunction with the ACP can also help to further simplify maintenance and renewal of domain certificates. Instead of relying on CRL, the lifetime of certificates can be made extremely small, for example in the order of hours. When a node fails to connect to the ACP within its certificate lifetime, it cannot connect to the ACP to renew its certificate across it (using just EST), but it can still renew its certificate as an "enrolled/expired pledge" via the BRSKI bootstrap proxy. This requires only that the BRSKI registrar honors expired domain certificates and that the pledge attempts to perform TLS authentication for BRSKI bootstrap using its expired domain certificate before falling back to attempting to use its IDevID certificate for BRSKI. This mechanism could also render CRLs unnecessary because the BRSKI registrar in conjunction with the CA would not renew revoked certificates – only a "Do-not-renew" list would be necessary on BRSKI registrars/CA.

In the absence of BRSKI or less secure variants thereof, provisioning of certificates may involve one or more touches or non-standardized automation. Node vendors usually support provisioning of certificates into nodes via PKCS#7 (see [RFC2315]) and may support this provisioning through vendor specific models via NETCONF ([RFC6241]). If such nodes also support NETCONF Zero-Touch ([RFC8572]) then this can be combined to zero-touch provisioning of domain certificates into nodes. Unless there are equivalent integration of NETCONF connections across the ACP as there is in BRSKI, this combination would not support zero-touch bootstrap across a not configured network though.

A.3. ACP Neighbor discovery protocol selection

This section discusses why GRASP DULL was chosen as the discovery protocol for L2 adjacent candidate ACP neighbors. The contenders considered where GRASP, mDNS or LLDP.

A.3.1. LLDP

LLDP and Cisco's earlier Cisco Discovery Protocol (CDP) are example of L2 discovery protocols that terminate their messages on L2 ports. If those protocols would be chosen for ACP neighbor discovery, ACP neighbor discovery would therefore also terminate on L2 ports. This would prevent ACP construction over non-ACP capable but LLDP or CDP enabled L2 switches. LLDP has extensions using different MAC addresses and this could have been an option for ACP discovery as well, but the additional required IEEE standardization and definition of a profile for such a modified instance of LLDP seemed to be more work than the benefit of "reusing the existing protocol" LLDP for this very simple purpose.

A.3.2. mDNS and L2 support

   Multicast DNNS (mDNS) [RFC6762] with DNS Service Discovery (DNS-SD)
   Resource Records (RRs) as defined in [RFC6763] is a key contender as
   an ACP discovery protocol. because it relies on link-local IP
   multicast, it does operates at the subnet level, and is also found in
   L2 switches. The authors of this document are not aware of mDNS
   implementation that terminate their mDNS messages on L2 ports instead
   of the subnet level. If mDNS was used as the ACP discovery mechanism
   on an ACP capable (L3)/L2 switch as outlined in Section 7, then this
   would be necessary to implement. It is likely that termination of
   mDNS messages could only be applied to all mDNS messages from such a
   port, which would then make it necessary to software forward any non-
   ACP related mDNS messages to maintain prior non-ACP mDNS
   functionality. Adding support for ACP into such L2 switches with
   mDNS could therefore create regression problems for prior mDNS
   functionality on those nodes. With low performance of software
   forwarding in many L2 switches, this could also make the ACP risky to
   support on such L2 switches.

A.3.3. Why DULL GRASP

   LLDP was not considered because of the above mentioned issues. mDNS
   was not selected because of the above L2 mDNS considerations and
   because of the following additional points:

   If mDNS was not already existing in a node, it would be more work to
   implement than DULL GRASP, and if an existing implementation of mDNS
   was used, it would likely be more code space than a separate
   implementation of DULL GRASP or a shared implementation of DULL GRASP
   and GRASP in the ACP.

A.4. Choice of routing protocol (RPL)

   This section motivates why RPL - "IPv6 Routing Protocol for Low-Power
   and Lossy Networks ([RFC6550] was chosen as the default (and in this
   specification only) routing protocol for the ACP. The choice and
   above explained profile was derived from a pre-standard
   implementation of ACP that was successfully deployed in operational
   networks.

   Requirements for routing in the ACP are:

* Self-management: The ACP must build automatically, without human
  intervention. Therefore, routing protocol must also work
  completely automatically. RPL is a simple, self-managing
  protocol, which does not require zones or areas; it is also self-
  configuring, since configuration is carried as part of the
  protocol (see Section 6.7.6 of [RFC6550]).
* Scale: The ACP builds over an entire domain, which could be a
  large enterprise or service provider network. The routing
  protocol must therefore support domains of 100,000 nodes or more,
  ideally without the need for zoning or separation into areas. RPL
  has this scale property. This is based on extensive use of
  default routing.
* Low resource consumption: The ACP supports traditional network
  infrastructure, thus runs in addition to traditional protocols.
  The ACP, and specifically the routing protocol must have low
  resource consumption both in terms of memory and CPU requirements.
  Specifically, at edge nodes, where memory and CPU are scarce,
  consumption should be minimal. RPL builds a DODAG, where the main
  resource consumption is at the root of the DODAG. The closer to
  the edge of the network, the less state needs to be maintained.
  This adapts nicely to the typical network design. Also, all
  changes below a common parent node are kept below that parent
  node.
* Support for unstructured address space: In the Autonomic
  Networking Infrastructure, node addresses are identifiers, and may
  not be assigned in a topological way. Also, nodes may move
  topologically, without changing their address. Therefore, the
  routing protocol must support completely unstructured address
  space. RPL is specifically made for mobile ad-hoc networks, with
  no assumptions on topologically aligned addressing.
* Modularity: To keep the initial implementation small, yet allow
  later for more complex methods, it is highly desirable that the
  routing protocol has a simple base functionality, but can import
  new functional modules if needed. RPL has this property with the
  concept of "objective function", which is a plugin to modify
  routing behavior.
* Extensibility: Since the Autonomic Networking Infrastructure is a
  new concept, it is likely that changes in the way of operation
  will happen over time. RPL allows for new objective functions to
  be introduced later, which allow changes to the way the routing
  protocol creates the DAGs.
* Multi-topology support: It may become necessary in the future to
  support more than one DODAG for different purposes, using
  different objective functions. RPL allow for the creation of
  several parallel DODAGs, should this be required. This could be
  used to create different topologies to reach different roots.

* No need for path optimization: RPL does not necessarily compute
  the optimal path between any two nodes. However, the ACP does not
  require this today, since it carries mainly non-delay-sensitive
  feedback loops. It is possible that different optimization
  schemes become necessary in the future, but RPL can be expanded
  (see point "Extensibility" above).

A.5. ACP Information Distribution and multicast

IP multicast is not used by the ACP because the ANI (Autonomic
Networking Infrastructure) itself does not require IP multicast but
only service announcement/discovery. Using IP multicast for that
would have made it necessary to develop a zero-touch auto configuring
solution for ASM (Any Source Multicast - the original form of IP
multicast defined in [RFC1112]), which would be quite complex and
difficult to justify. One aspect of complexity where no attempt at a
solution has been described in IETF documents is the automatic-
selection of routers that should be PIM Sparse Mode (PIM-SM)
Rendezvous Points (RPs) (see [RFC7761]). The other aspects of
complexity are the implementation of MLD ([RFC4604]), PIM-SM and
Anycast-RP (see [RFC4610]). If those implementations already exist
in a product, then they would be very likely tied to accelerated
forwarding which consumes hardware resources, and that in return is
difficult to justify as a cost of performing only service discovery.

Some future ASA may need high performance in-network data
replication. That is the case when the use of IP multicast is
justified. Such an ASA can then use service discovery from ACP
GRASP, and then they do not need ASM but only SSM (Source Specific
Multicast, see [RFC4607]) for the IP multicast replication. SSM
itself can simply be enabled in the Data-Plane (or even in an update
to the ACP) without any other configuration than just enabling it on
all nodes and only requires a simpler version of MLD (see [RFC5790]).

LSP (Link State Protocol) based IGP routing protocols typically have
a mechanism to flood information, and such a mechanism could be used
to flood GRASP objectives by defining them to be information of that
IGP. This would be a possible optimization in future variations of
the ACP that do use an LSP routing protocol. Note though that such a
mechanism would not work easily for GRASP M_DISCOVERY messages which
are intelligently (constrained) flooded not across the whole ACP, but
only up to a node where a responder is found. We do expect that many
future services in ASA will have only few consuming ASA, and for
those cases, M_DISCOVERY is the more efficient method than flooding
across the whole domain.

Because the ACP uses RPL, one desirable future extension is to use
RPLs existing notion of DODAG, which are loop-free distribution
trees, to make GRASP flooding more efficient both for M_FLOOD and
M_DISCOVERY. See Section 6.13.5 how this will be specifically
beneficial when using NBMA interfaces. This is not currently
specified in this document because it is not quite clear yet what
exactly the implications are to make GRASP flooding depend on RPL
DODAG convergence and how difficult it would be to let GRASP flooding
access the DODAG information.

A.6. CAs, domains and routing subdomains

There is a wide range of setting up different ACP solution by
appropriately using CAs and the domain and rsub elements in the acp-
node-name in the domain certificate. We summarize these options here
as they have been explained in different parts of the document in
before and discuss possible and desirable extensions:

An ACP domain is the set of all ACP nodes that can authenticate each
other as belonging to the same ACP network using the ACP domain
membership check (Section 6.2.3). GRASP inside the ACP is run across
all transitively connected ACP nodes in a domain.

The rsub element in the acp-node-name permits the use of addresses
from different ULA prefixes. One use case is to create multiple
physical networks that initially may be separated with one ACP domain
but different routing subdomains, so that all nodes can mutual trust
their ACP certificates (not depending on rsub) and so that they could
connect later together into a contiguous ACP network.

One instance of such a use case is an ACP for regions interconnected
via a non-ACP enabled core, for example due to the absence of product
support for ACP on the core nodes. ACP connect configurations as
defined in this document can be used to extend and interconnect those
ACP islands to the NOC and merge them into a single ACP when later
that product support gap is closed.

Note that RPL scales very well. It is not necessary to use multiple
routing subdomains to scale ACP domains in a way that would be
required if other routing protocols where used. They exist only as
options for the above mentioned reasons.

If different ACP domains are to be created that should not allow to
connect to each other by default, these ACP domains simply need to
have different domain elements in the acp-node-name. These domain
elements can be arbitrary, including subdomains of one another:
Domains "example.com" and "research.example.com" are separate domains
if both are domain elements in the acp-node-name of certificates.

It is not necessary to have a separate CA for different ACP domains:
an operator can use a single CA to sign certificates for multiple ACP
domains that are not allowed to connect to each other because the
checks for ACP adjacencies includes comparison of the domain part.

If multiple independent networks choose the same domain name but had
their own CA, these would not form a single ACP domain because of CA
mismatch. Therefore, there is no problem in choosing domain names
that are potentially also used by others. Nevertheless it is highly
recommended to use domain names that one can have high probability to
be unique. It is recommended to use domain names that start with a
DNS domain names owned by the assigning organization and unique
within it. For example, "acp.example.com" if you own "example.com".

A.7. Intent for the ACP

Intent is the architecture component of autonomic networks according
to [I-D.ietf-anima-reference-model] that allows operators to issue
policies to the network. Its applicability for use is quite flexible
and freeform, with potential applications including policies flooded
across ACP GRASP and interpreted on every ACP node.

One concern for future definitions of Intent solutions is the problem
of circular dependencies when expressing Intent policies about the
ACP itself.

For example, Intent could indicate the desire to build an ACP across
all domains that have a common parent domain (without relying on the
rsub/routing-subdomain solution defined in this document). For
example, ACP nodes with domain "example.com", "access.example.com",
"core.example.com" and "city.core.example.com" should all establish
one single ACP.

If each domain has its own source of Intent, then the Intent would
simply have to allow adding the peer domains TA and domain names to
the parameters for the ACP domain membership check (Section 6.2.3) so
that nodes from those other domains are accepted as ACP peers.

If this Intent was to be originated only from one domain, it could
likely not be made to work because the other domains will not build
any ACP connection amongst each other, whether they use the same or
different CA due to the ACP domain membership check.

If the domains use the same CA one could change the ACP setup to permit for the ACP to be established between two ACP nodes with different acp-domain-names, but only for the purpose of disseminating limited information, such as Intent, but not to set up full ACP connectivity, specifically not RPL routing and passing of arbitrary GRASP information. Unless the Intent policies permit this to happen across domain boundaries.

This type of approach where the ACP first allows Intent to operate and only then sets up the rest of ACP connectivity based on Intent policy could also be used to enable Intent policies that would limit functionality across the ACP inside a domain, as long as no policy would disturb the distribution of Intent. For example, to limit reachability across the ACP to certain type of nodes or locations of nodes.

A.8. Adopting ACP concepts for other environments

The ACP as specified in this document is very explicit about the choice of options to allow interoperable implementations. The choices made may not be the best for all environments, but the concepts used by the ACP can be used to build derived solutions:

The ACP specifies the use of ULA and deriving its prefix from the domain name so that no address allocation is required to deploy the ACP. The ACP will equally work not using ULA but any other /48 IPv6 prefix. This prefix could simply be a configuration of the ACP registrars (for example when using BRSKI) to enroll the domain certificates - instead of the ACP registrar deriving the /48 ULA prefix from the AN domain name.

Some solutions may already have an auto-addressing scheme, for example derived from existing unique device identifiers (e.g., MAC addresses). In those cases it may not be desirable to assign addresses to devices via the ACP address information field in the way described in this document. The certificate may simply serve to identify the ACP domain, and the address field could be omitted. The only fix required in the remaining way the ACP operate is to define another element in the domain certificate for the two peers to decide who is the Decider and who is the Follower during secure channel building. Note though that future work may leverage the acp address to authenticate "ownership" of the address by the device. If the address used by a device is derived from some pre-existing permanent local ID (such as MAC address), then it would be useful to store that address in the certificate using the format of the access address information field or in a similar way.

The ACP is defined as a separate VRF because it intends to support
well managed networks with a wide variety of configurations.
Therefore, reliable, configuration-indestructible connectivity cannot
be achieved from the Data-Plane itself. In solutions where all
transit connectivity impacting functions are fully automated
(including security), indestructible and resilient, it would be
possible to eliminate the need for the ACP to be a separate VRF.
Consider the most simple example system in which there is no separate
Data-Plane, but the ACP is the Data-Plane. Add BRSKI, and it becomes
a fully autonomic network - except that it does not support automatic
addressing for user equipment. This gap can then be closed for
example by adding a solution derived from
[I-D.ietf-anima-prefix-management].

TCP/TLS as the protocols to provide reliability and security to GRASP
in the ACP may not be the preferred choice in constrained networks.
For example, CoAP/DTLS (Constrained Application Protocol) may be
preferred where they are already used, allowing to reduce the
additional code space footprint for the ACP on those devices. Hop-
by-hop reliability for ACP GRASP messages could be made to support
protocols like DTLS by adding the same type of negotiation as defined
in this document for ACP secure channel protocol negotiation. End-
to-end GRASP connections can be made to select their transport
protocol in future extensions of the ACP meant to better support
constrained devices by indicating the supported transport protocols
(e.g. TLS/DTLS) via GRASP parameters of the GRASP objective through
which the transport endpoint is discovered.

The routing protocol RPL used for the ACP does explicitly not
optimize for shortest paths and fastest convergence. Variations of
the ACP may want to use a different routing protocol or introduce
more advanced RPL profiles.

Variations such as what routing protocol to use, or whether to
instantiate an ACP in a VRF or (as suggested above) as the actual
Data-Plane, can be automatically chosen in implementations built to
support multiple options by deriving them from future parameters in
the certificate. Parameters in certificates should be limited to
those that would not need to be changed more often than certificates
would need to be updated anyhow; Or by ensuring that these parameters
can be provisioned before the variation of an ACP is activated in a
node. Using BRSKI, this could be done for example as additional
follow-up signaling directly after the certificate enrollment, still
leveraging the BRSKI TLS connection and therefore not introducing any
additional connectivity requirements.

Last but not least, secure channel protocols including their
encapsulations are easily added to ACP solutions. ACP hop-by-hop
network layer secure channels could also be replaced by end-to-end
security plus other means for infrastructure protection. Any future
network OAM should always use end-to-end security anyhow and can
leverage the domain certificates and is therefore not dependent on
security to be provided for by ACP secure channels.

A.9. Further (future) options

A.9.1. Auto-aggregation of routes

   Routing in the ACP according to this specification only leverages the
   standard RPL mechanism of route optimization, e.g. keeping only
   routes that are not towards the RPL root. This is known to scale to
   networks with 20,000 or more nodes. There is no auto-aggregation of
   routes for /48 ULA prefixes (when using rsub in the acp-node-name)
   and/or Zone-ID based prefixes.

   Automatic assignment of Zone-ID and auto-aggregation of routes could
   be achieved for example by configuring zone-boundaries, announcing
   via GRASP into the zones the zone parameters (zone-ID and /48 ULA
   prefix) and auto-aggregating routes on the zone-boundaries. Nodes
   would assign their Zone-ID and potentially even /48 prefix based on
   the GRASP announcements.

A.9.2. More options for avoiding IPv6 Data-Plane dependencies

   As described in Section 6.13.2, the ACP depends on the Data-Plane to
   establish IPv6 link-local addressing on interfaces. Using a separate
   MAC address for the ACP allows to fully isolate the ACP from the
   Data-Plane in a way that is compatible with this specification. It
   is also an ideal option when using Single-root input/output
   virtualization (SR-IOV - see https://en.wikipedia.org/wiki/Single-
   root_input/output_virtualization) in an implementation to isolate the
   ACP because different SR-IOV interfaces use different MAC addresses.

   When additional MAC address(es) are not available, separation of the
   ACP could be done at different demux points. The same subnet
   interface could have a separate IPv6 interface for the ACP and Data-
   Plane and therefore separate link-local addresses for both, where the
   ACP interface is non-configurable on the Data-Plane. This too would
   be compatible with this specification and not impact
   interoperability.

   An option that would require additional specification is to use a
   different Ethertype from 0x86DD (IPv6) to encapsulate IPv6 packets
   for the ACP. This would be a similar approach as used for IP

authentication packets in [IEEE-802.1X] which use the Extensible
Authentication Protocol over Local Area Network (EAPoL) ethertype
(0x88A2).

Note that in the case of ANI nodes, all the above considerations
equally apply to the encapsulation of BRSKI packets including GRASP
used for BRSKI.

A.9.3. ACP APIs and operational models (YANG)

Future work should define YANG ([RFC7950]) data model and/or node
internal APIs to monitor and manage the ACP.

Support for the ACP Adjacency Table (Section 6.3) and ACP GRASP need
to be included into such model/API.

A.9.4. RPL enhancements

```
      ..... USA ......              ..... Europe ......

         NOC1                          NOC2
          |                            |
          |            metric 100      |
         ACP1 -------------------------- ACP2   .
          |                            |        . WAN
          |  metric 10      metric 20  |        . Core
          |                            |        .
         ACP3 -------------------------- ACP4   .
          |            metric 100      |
          |                            |        .
          |                            |        . Sites
         ACP10                         ACP11  .
```
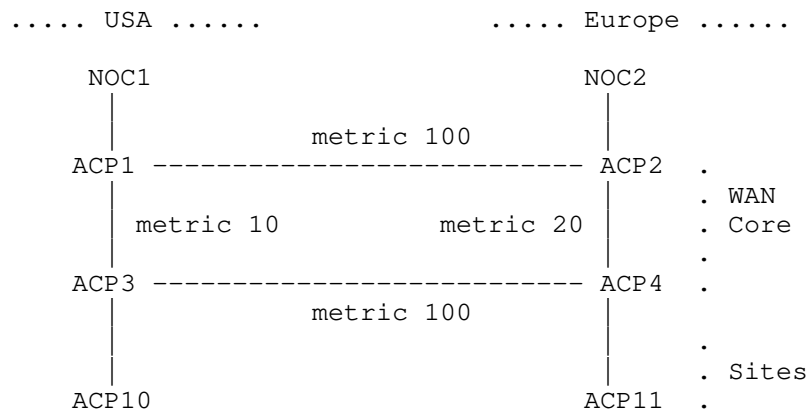
                      Figure 19: Dual NOC

The profile for RPL specified in this document builds only one
spanning-tree path set to a root, typically a registrar in one NOC.
In the presence of multiple NOCs, routing toward the non-root NOCs
may be suboptimal. Figure 19 shows an extreme example. Assuming
that node ACP1 becomes the RPL root, traffic between ACP11 and NOC2
will pass through ACP4-ACP3-ACP1-ACP2 instead of ACP4-ACP2 because
the RPL calculated DODAG/routes are shortest paths towards the RPL
root.

To overcome these limitations, extensions/modifications to the RPL
profile can provide optimality for multiple NOCs. This requires
utilizing Data-Plane artifact including IPinIP encap/decap on ACP
routers and processing of IPv6 RPI headers. Alternatively, (Src,Dst)
routing table entries could be used.

Flooding of ACP GRASP messages can be further constrained and
therefore optimized by flooding only via links that are part of the
RPL DODAG.

A.9.5. Role assignments

ACP connect is an explicit mechanism to "leak" ACP traffic explicitly
(for example in a NOC). It is therefore also a possible security gap
when it is easy to enable ACP connect on arbitrary compromised ACP
nodes.

One simple solution is to define an extension in the ACP certificates
ACP information field indicating the permission for ACP connect to be
configured on that ACP node. This could similarly be done to decide
whether a node is permitted to be a registrar or not.

Tying the permitted "roles" of an ACP node to the ACP certificate
provides fairly strong protection against misconfiguration, but is
still subject to code modifications.

Another interesting role to assign to certificates is that of a NOC
node. This would allow to limit certain type of connections such as
OAM TLS connections to only NOC initiator or responders.

A.9.6. Autonomic L3 transit

In this specification, the ACP can only establish autonomic
connectivity across L2 hops and only explicitly configured options to
tunnel across L3. Future work should specify mechanisms to
automatically tunnel ACP across L3 networks. A hub&spoke option
would allow to tunnel across the Internet to a cloud or central
instance of the ACP, a peer-to-peer tunneling mechanism could tunnel
ACP islands across an L3VPN infrastructure.

A.9.7. Diagnostics

Section 9.1 describes diagnostics options that can be done without
changing the external, interoperability affecting characteristics of
ACP implementations.

Even better diagnostics of ACP operations is possible with additional
signaling extensions, such as:

   1. Consider if LLDP should be a recommended functionality for ANI
       devices to improve diagnostics, and if so, which information
       elements it should signal (noting that such information is
       conveyed in an insecure manner). Includes potentially new
       information elements.
   2. In alternative to LLDP, A DULL GRASP diagnostics objective could
       be defined to carry these information elements.
   3. The IDevID certificate of BRSKI pledges should be included in the
       selected insecure diagnostics option. This may be undesirable
       when exposure of device information is seen as too much of a
       security issue (ability to deduce possible attack vectors from
       device model for example).
   4. A richer set of diagnostics information should be made available
       via the secured ACP channels, using either single-hop GRASP or
       network wide "topology discovery" mechanisms.

A.9.8. Avoiding and dealing with compromised ACP nodes

   Compromised ACP nodes pose the biggest risk to the operations of the
   network. The most common type of compromise is leakage of
   credentials to manage/configure the device and the application of
   malicious configuration including the change of access credentials,
   but not the change of software. Most of today's networking equipment
   should have secure boot/software infrastructure anyhow, so attacks
   that introduce malicious software should be a lot harder.

   The most important aspect of security design against these type of
   attacks is to eliminate password based configuration access methods
   and instead rely on certificate based credentials handed out only to
   nodes where it is clear that the private keys cannot leak. This
   limits unexpected propagation of credentials.

   If password based credentials to configure devices still need to be
   supported, they must not be locally configurable, but only be
   remotely provisioned or verified (through protocols like RADIUS or
   Diameter), and there must be no local configuration permitting to
   change these authentication mechanisms, but ideally they should be
   autoconfiguring across the ACP. See
   [I-D.eckert-anima-noc-autoconfig].

   Without physical access to the compromised device, attackers with
   access to configuration should not be able to break the ACP
   connectivity, even when they can break or otherwise manipulate
   (spoof) the Data-Plane connectivity through configuration. To
   achieve this, it is necessary to avoid providing configuration
   options for the ACP, such as enabling/disabling it on interfaces.
   For example, there could be an ACP configuration that locks down the
   current ACP config unless factory reset is done.

With such means, the valid administration has the best chances to
maintain access to ACP nodes, discover malicious configuration though
ongoing configuration tracking from central locations for example,
and to react accordingly.

The primary reaction is withdrawal/change of credentials, terminate
malicious existing management sessions and fixing the configuration.
Ensuring that management sessions using invalidated credentials are
terminated automatically without recourse will likely require new
work.

Only when these steps are not feasible would it be necessary to
revoke or expire the ACP certificate credentials and consider the
node kicked off the network - until the situation can be further
rectified, likely requiring direct physical access to the node.

Without extensions, compromised ACP nodes can only be removed from
the ACP at the speed of CRL/OCSP information refresh or expiry (and
non-removal) of short lived certificates. Future extensions to the
ACP could for example use GRASP flooding distribution of triggered
updates of CRL/OCSP or explicit removal indication of the compromised
nodes domain certificate.

A.9.9. Detecting ACP secure channel downgrade attacks

The following text proposes a mechanism to protect against downgrade
attacks without introducing a new specialized UPFRONT GRASP secure
channel mechanism. Instead, it relies on running GRASP after
establishing a secure channel protocol to verify if the established
secure channel option could have been the result of a MITM downgrade
attack:

MITM attackers can force downgrade attacks for ACP secure channel
selection by filtering/modifying DULL GRASP messages and/or actual
secure channel data packets. For example, if at some point in time
DTLS traffic could be easier decrypted than traffic of IKEv2, the
MITM could filter all IKEv2 packets to force ACP nodes to use DTLS
(assuming the ACP nodes in question supported both DTLS and IKEv2).

For cases where such MITM attacks are not capable to inject malicious
traffic (but only to decrypt the traffic), a downgrade attack could
be discovered after a secure channel connection is established, for
example by use of the following type of mechanism:

After the secure channel connection is established, the two ACP peers
negotiate via an appropriate (To Be Defined) GRASP negotiation which
ACP secure channel protocol should have been selected between them
(in the absence of a MITM attacker). This negotiation would have to

signal the DULL GRASP announced ACP secure channel options by each
peer followed by an announcement of the preferred secure channel
protocol by the ACP peer that is the Decider in the secure channel
setup, e.g. the ACP peer that is deciding which secure channel
protocol to pick. If that chosen secure channel protocol is
different from the one that actually was chosen, then this mismatch
is an indication that there is a MITM attacker or other similar issue
(firewall prohibiting the use of specific protocols) that caused a
non-preferred secure channel protocol to be chosen. This discovery
could then result in mitigation options such as logging and ensuing
investigations.

Appendix B. Unfinished considerations (To Be Removed From RFC)

   [RFC-Editor: This whole appendix B. and its subsections to be removed
   for the RFC.

   This appendix contains unfinished considerations that are removed
   from the RFC, they are maintained in this draft as a log of the state
   of discussion and point of reference. Together with this appendix,
   also the references pointing to it are marked to be removed from the
   RFC because no consensus could be reached that a self-reference to a
   draft version of the RFC is an appropriate breadcrumb to point to
   unfinished considerations.

   The authors plan to move these considerations into a new target
   informational draft, please look for draft-eckert-anima-acp-
   considerations.

B.1. Considerations for improving secure channel negotiation

   Proposed text from Benjamin Kaduk. It is suggested to replace the
   text of appendix A.6 in previous versions of this draft (up to
   version 29).

   The discovery procedure in this specification for low-level ACP
   channel support by layer-2 peers involves DULL GRASP and attempting
   (usually in parallel) to establish all supported channel types,
   learning the peer ACP address and correspondingly the assignment of
   Decider and Follower roles, and tearing down all channels other than
   the one preferred by the Decider. This procedure, in general,
   becomes resource intensive as the number of possible secure channels
   grows; even worse, under some threat models, the security of the
   discovery result is only as strong as the weakest supported secure
   channel protocol. Furthermore, the unilateral determination of
   "best" channel type by the Decider does not result in the optimal
   outcome in all possible scenarios.

This situation is tolerable at present, with only two secure channels (DTLS and IPsec) defined, but long-term agility in the vein of [BCP201] will require the introduction of an alternate discovery/ negotiation procedure. While IKEv2 is the IETF standard protocol for negotiating security associations, it currently does not have a defined mechanism flexible enough to negotiate the parameters needed for, e.g., an ACP DTLS channel, let alone for allowing ACP peers to indicate their preference metrics for channel selection. Such a mechanism or mechanisms could be defined, but if ACP agility requires introducing a new channel type, for example MacSec, IKEv2 would again need to be extended in order to negotiate an ACP MacSec association. Making ACP channel agility dependent on updates to IKEv2 is likely to result in obstacles due to different timescales of evolution, since IKEv2 implementations help form the core of Internet-scale security infrastructure and must accordingly be robust and thoroughly tested.

Accordingly, a dedicated ACP channel negotiation mechanism is appropriate as a way to provide long-term algorithm and secure- channel protocol agility. Such a mechanism is not currently defined, but one possible design is as follows. A new DULL GRASP objective is defined to indicate the GRASP-over-TLS channel, which is by definition preferred to other channel types (including DTLS and IPsec). When both peers advertise support for GRASP-over-TLS, GRASP- over-TLS must run to completion before other channel types are attempted. The GRASP-over-TLS channel performs the necessary negotiation by establishing a TLS connection between the peers and using that connection to secure a dedicated GRASP instance for negotiating supported channel types and preference metrics. This provides a rich language for determining what secure channel protocol to use for the ACP link while taking into account the capabilities and preferences of the ACP peers, all protected by the security of the TLS channel.

B.2. ACP address verification

The AcpNodeName of most ACP nodes contains in the acp-address field the primary ACP address to be used by the node for end-to-end connections across ACP secure channels. Nevertheless, there is no verification of an ACP peers address specified in this document. This section explains the current understanding as to why this is not done.

Not all ACP nodes will have an actual IPv6 address in the acp-address field of their AcpNodeName. Those who do not include nodes that do not support ACP secure channels, such as pre-existing NOC equipment that only connects to the ACP via ACP connect interfaces. Likewise, future ACP node type that may want to have their Node-ID not be defined by an ACP registrar, but differently cannot have the ACP

address be provided in their ACP certificate where it would be
defined by the registrar. In result, any scheme that would rely on
verification of the acp-address in the ACP certificate would only
apply to a subset of ACP nodes.

The transport stack network layer address used for ACP secure
channels is not the acp-address. For automatically established ACP
secure channels, it is a link-local IPv6 address. For explicitly
configured ACP secure channels (to reach across non ACP L3 network
segments), the address is any IPv4 or IPv6 address routable to that
remote destination.

When the acp-address is actually used across the ACP, it can only be
verified by a peer when the peer has the certificate of the peer.
Unless further higher layer mechanisms are developed on top of the
ACP (for example via ASA), the only mechanism to access a peers ACP
certificate is for secure connections in which the peers certificates
are exchanged and cryptographically verified, e.g. TLS and DTLS.
Initially, it is expected that the ACP will carry many legacy network
management control connections that unfortunately not end-to-end
authenticated but that are solely protected by being carried across
the ACP secure channels. ACP address verification therefore cannot
be used for such connections without additional higher layer
components.

For the remaining (TLS/DTLS) connections for which address
verification can be used, the main question is: what additional
benefit would address verification provide?

The main value that transport stack network layer address
verification could provide for these type of connections would be the
discovery of on-path transport proxies. For example, in case of
BRSKI, pledges connect to an ACP registrar via an ASA implementing a
TCP proxy because the pledge itself has at that point in time no ACP
certificate valid to build ACP secure channels and hence needs to
rely on such a proxy. This is one example where such a TCP proxy is
required and not a form of attack.

In general, on path TCP proxies could be a form of attack, but it
stands to reason, that an attacker that manages to enable a malicious
TCP proxy could likely equally build a transparent proxy not changing
the network layer addresses. Only when the attacker operates off-
path would this option not be possible. Such attacks could indeed be
possible: An impaired ACP node could announce itself as another
service instance for a service whose utilization it wants to attack.
It could then attempt to look like a valid server by simply TCP
proxying the clients connections to a valid server and then attack
the connections passing through it (passive decrypting or passive

fingerprint analysis). But like the BRSKI proxy, this behavior could
be perfectly legitimate and not an attack. For example, TCP has in
the past often suffered from performance issues across difficult
(high capacity, high loss) paths, and TCP proxies where and are being
used simply as a tool for isolating such path segments (such as a
WAN), and providing caching and local-retransmit of in-transit
packets, reducing the effective path segment capacity.

As explained elsewhere in this document already, considerations for
these type of attack are therefore outside the scope of the ACP but
fundametal to further design of the ASA infrastructure. Beyond
distinguishing whether a TCP proxy would be beneficial or malicious,
the even more fundamental question is how to determine from a
multitude of service announcements which instance is the most
trustworthy and functionally best. In the Internet/web, this
question is NOT solved inside the network but through off-net human
interaction ("trust me, the best search engine is www.<insert-your-
personal-recommendation>.com").

B.3. Public CA considerations

Public CAs are outside the scope of this document for the following
reasons. This appendix describes the current state of understanding
for those interested to consider utilizing public CA for the ACP in
the future.

If public CA where to be used to enroll ACP nodes and act as TA, this
would require a model in which the public CA would be able to assert
the ownership of the information requested in the certificate,
especially the AcpNodeName, for example mitigated by the domain
registrar(s). Due to the use of a new, ACP unique encoding of the
AcpNodeName, there is no mechanism for public CA to do so. More
importantly though, isolation between ACPs of disjoint operated ACPs
is achieved in the current ACP design through disjoint TA. A public
CA is in general based on a single (set of) TA shared across all
certificates signed by the CA.

Due to the fact that the ACP domain membership check also validates
that a peers domain name in the AcpNodeName matches that of the ACP
node itself, it would be possible to use the same TA across disjoint
ACP domains, but the security and attack implications of such an
approach are beyond the scope of this document.

The use of ULA addresses in the AcpNodeName is another novel aspect
for certificates from a possible public CA. Typically, ULA addresses
are not meant to be signed by a public CA when carried in an address
field, because there is no ownership of a particular ULA address in
the scope of the Internet, which is what public CA operate on.

Nevertheless, the ULA addresses used by the ACP are scoped to be
valid only within the confines of a specific ACP as defined by the
domain name in the AcpNodeName. However, this understanding has not
been reviewed or accepted by any bodies responsible for policies of
public CA.

Because in this specification, ACPs are isolated from each other
primarily by their TA, when a public CA would intend to sign ACP
certificates and using a single TA to sign TA of ACP certificates
from different operators/domain, it could do so by ensuring that the
domain name in the AcpNodeName was a globally owned DNS ACP domain
name of the organization, and beyond that, it would need to validate
that the ACP registrar of that domain who is mitigating the
enrollment is authorized to vouch for the ownership of the acp-
address within the scope of the ACP domain name.

B.4. Hardening DULL GRASP considerations

DULL GRASP suffers from similar type of DoS attacks as many link-
local multicast discovery protocols, for example mDNS. Attackers on
a subnet may be able to inject malicious DULL GRASP messages that are
indistinguishable from non-malicious DULL GRASP messages to create
Denial-of-Service (DoS) attacks that force ACP nodes to attempt many
unsuccessful ACP secure channel connections.

When an ACP node sees multiple AN_ACP objectives for the same secure
channel protocol on different transport addresses, it could prefer
connecting via the well-known transport address if the secure channel
method has one, such as UDP port 500 for IKEv2. For protocols such
as (ACP secure channel over) DTLS for which there are no well defined
port number, this heuristic does not provide benefits though.

DoS attack with port numbers can also be eliminated by relying on
well known-port numbers implied by the GRASP method-name. For
example, a future service name of "DTLSacp" could be defined to be
associated only to a newly to be assigned well known UDP port for ACP
over DTLS, and the port number in the GRASP transport address
information would be ignored. Note that there is already a variety
of ports assigned to specific protocols over DTLS by IANA, so
especially for DTLS this would not be uncommon.

Authors' Addresses

Toerless Eckert (editor)
Futurewei Technologies Inc. USA
2330 Central Expy
Santa Clara, 95050
United States of America

Email: tte+ietf@cs.fau.de

Michael H. Behringer (editor)

Email: michael.h.behringer@gmail.com

Steinthor Bjarnason
Arbor Networks
2727 South State Street, Suite 200
Ann Arbor, MI 48104
United States

Email: sbjarnason@arbor.net