Abstract

In certain scenarios, for example when bootstrapping a network, it is desirable to automatically bring up a secure, routed control plane, which is independent of device configurations and global routing table. This document describes an approach for a logically separated "Autonomic Control Plane", which can be used as a "virtual out of band channel" - a self-managing overlay network, which is independent of configuration, addressing and routing on the data plane.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 7, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Today, the management and control plane of networks typically runs in the global routing table, which is dependent on correct configuration and routing. Misconfigurations or routing problems can therefore disrupt management and control channels. Traditionally, an out of band network has been used to recover from such problems, or personnel is sent on site to access devices through console ports. However, both options are operationally expensive.

In increasingly automated networks either controllers or distributed autonomic service agents in the network require a control plane which
is independent of the network they manage, to avoid impacting their own operations.

This document describes a self-forming, self-managing and self-protecting "Autonomic Control Plane" (ACP) which is inband on the network, yet independent of configuration, addressing and routing problems (for details how this achieved, see Section 4). It therefore remains operational even in the presence of configuration errors, addressing or routing issues, or where policy could inadvertently affect control plane connectivity. The Autonomic Control Plane serves several purposes at the same time:

- An operator can use it to log into remote devices, even if the data plane is misconfigured or unconfigured.
- A controller or network management system can use it to securely bootstrap network devices in remote locations, even if the 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.pritikin-anima-bootstrapping-keyinfra]
- Devices can use the ACP for direct decentralised communications, such as negotiations or discovery. The ACP therefore supports directly Autonomic Networking functions, as described in [I-D.behringer-anima-reference-model]. For example, GDNP [I-D.carpenter-anima-gdn-protocol] can run inside the ACP.

The Autonomic Control plane relies exclusively on IPv6 for its operation, and all operations in the ACP are exclusively IPv6. Since the ACP is a new approach, there should be no need to support dual stack IPv4/v6. The network operator can configure the network data plane for any protocol, including IPv4 or IPv6.

This document describes how the Autonomic Control Plane is constructed, and some use cases for it. The document "Autonomic Network Stable Connectivity" [I-D.eckert-anima-stable-connectivity] describes how the ACP can be used to provide stable connectivity for OAM applications. It also explains on how existing management solutions can leverage the ACP in parallel with traditional management models, when to use the ACP versus the data plane, how to integrate IPv4 based management, etc.

The ACP can support Autonomic Networking functions. For background information, definitions and design goals of Autonomic Networking, refer to [I-D.irtf-nmrg-autonomic-network-definitions]. For a gap analysis please see [I-D.irtf-nmrg-an-gap-analysis].
2. Use Cases for an Autonomic Control Plane

2.1. Secure Bootstrap over an Unconfigured Network

Today, bootstrapping a new device typically requires all devices between a controlling node (such as an SDN controller) and the new device to be completely and correctly addressed, configured and secured. Therefore, bootstrapping a network happens in layers around the controller. 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 between.

With the ACP, secure bootstrap of new devices can happen without requiring any configuration on the network. A new device can automatically be bootstrapped in a secure fashion and be deployed with a domain certificate. This does not require any configuration on intermediate nodes, because they can communicate through the ACP.

2.2. Data Plane Independent Permanent Reachability

Today, most critical control plane protocols and network management protocols are running in the data plane (global routing table) of the network. This leads to undesirable dependencies between control and management plane on one side and the data plane on the other: Only if the data plane is operational, will the other planes work as expected.

Data plane connectivity can be affected by errors and faults, for example certain AAA misconfigurations 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 console access can help recover from such issues.

Data plane dependencies also affect NOC/SDN controller applications: Certain network changes are today hard to operate, 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.

The ACP provides reachability that is largely independent of the data plane, which allows control plane and management 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 devices regardless of their configuration or global routing table.
o For control plane protocols, the ACP allows their operation even when the data plane is temporarily faulty, or during transitional events, such as routing changes, which may affect the control plane at least temporarily. This is specifically important for autonomic service agents, which could affect data plane connectivity.

The document "Autonomic Network Stable Connectivity" [I-D.eckert-anima-stable-connectivity] explains the use cases for the ACP in significantly more detail and explains how the ACP can be used in practical network operations.

3. Overview

The Autonomic Control Plane is constructed in the following way (for details, see Section 4):

o Each autonomic node creates a virtual routing and forwarding (VRF) instance, or a similar virtual context.

o When an autonomic node discovers another autonomic node from the same domain, it authenticates that node and negotiates a secure tunnel to it. These tunnels are placed into the previously set up VRF. This creates an overlay network with hop-by-hop tunnels.

o Inside the ACP VRF, each node sets up a loopback interface with a ULA IPv6 address.

o Each node runs a lightweight routing protocol, to announce reachability of the loopback addresses inside the ACP.

o NMS systems or controllers have to be manually connected into the ACP.

o None of the above operations is reflected in the configuration of the device.

The following figure illustrates the ACP.
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. This way the ACP can be built even if there is no configuration on the devices, or if the data plane has issues such as addressing or routing problems.

4. Self-Creation of an Autonomic Control Plane

This section describes the steps to set up an Autonomic Control Plane, and highlights the key properties which make it "indestructible" against many inadvert changes to the data plane, for example caused by misconfigurations.

4.1. Preconditions

Each autonomic device has a globally unique domain certificate, with which it can cryptographically assert its membership of the domain. The document [I-D.pritikin-anima-bootstrapping-keyinfra] describes how a domain certificate can be automatically and securely derived from a vendor specific Unique Device Identifier (UDI) or IDevID certificate. (Note the UDI used in this document is NOT the UUID specified in [RFC4122].)

4.2. Adjacency Discovery

Adjacency discovery exchanges identity information about neighbors, either the UDI or, if present, the domain certificate (see Section 4.1. This document assumes the existence of a domain certificate.
Adjacency discovery provides a table of information of adjacent neighbors. Each neighbor is identified by a globally unique device identifier (UDI).

The adjacency table contains the following information about the adjacent neighbors.

- Globally valid Unique device identifier (UDI).
- Link Local IPv6 address with its scope.
- Trust information: The certificate chain, if available.
- Validity of the trust (once validated, see next section).

Adjacency discovery can populate this table by several means. One such mechanism is to discover using link local multicast probes, which has no dependency on configured addressing and is preferable in an autonomic network.

The "Generic Discovery and Negotiation Protocol" GDNP described in [I-D.carpenter-anima-gdn-protocol] is a possible candidate protocol to meet the requirements for Adjacency Discovery described here.

4.3. Authenticating Neighbors

Each neighbor in the adjacency table is authenticated. The result of the authentication of the neighbor information is stored in the adjacency table. We distinguish the following cases:

- Inside the domain: If the domain certificate presented is validated (including proof of possession of the corresponding private key) to be in the same domain as that of the autonomic entity then the neighbor is deemed to be inside the autonomic domain. Only entities inside the autonomic domain will by default be able to establish the autonomic control plane. Alternatively, policy can define whether to simply trust devices with the same trust anchor. An ACP channel will be established.

- Outside the domain: If there is no domain certificate presented by the neighbor, or if the domain certificate presented is invalid or expired, then the neighbor is deemed to be outside the autonomic domain. No ACP channel will be established.

Certificate management questions such as enrolment, revocation, renewal, etc, are not discussed in this draft. Please refer to [I-D.pritikin-anima-bootstrapping-keyinfra] for more details.
4.4. Capability Negotiation

Autonomic devices have different capabilities based on the type of device and where it is deployed. To establish a trusted secure communication channel, devices must be able to negotiate with each neighbor a set of parameters for establishing the communication channel, most notably channel type and security type. The channel type could be any tunnel mechanism that is feasible between two adjacent neighbors, for example a GRE tunnel. The security type could be any of the channel protection mechanism that is available between two adjacent neighbors on a given channel type, for example TLS, DTLS or IPsec. The establishment of the autonomic control plane can happen after the channel type and security type is negotiated.

The "Generic Discovery and Negotiation Protocol GDNP described in [I-D.carpenter-anima-gdn-protocol] is a possible candidate protocol to meet the requirements for capability negotiation described here.

4.5. Channel Establishment

After authentication and capability negotiation autonomic nodes establish a secure channel towards their direct AN neighbors with the above negotiated parameters. In order to be independent of configured link addresses, these channels can be implemented in several ways:

- As a secure IP tunnel (e.g., IPsec, DTLS, TLS, etc.), using IPv6 link local addresses between two adjacent neighbors. This way, the ACP tunnels are independent of correct network wide routing. They also do not require larger than link local scope addresses, which would normally need to be configured or maintained. Each AN node MUST support this function.

- L2 separation, for example via a separate 802.1q tag for ACP traffic. This even further reduces dependency against the data plane (not even IPv6 link-local there required), but may be harder to implement.

Since channels are 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 4.8.

If two nodes are connected via several links, the ACP SHOULD be established on 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 devices, because state needs to
be kept per ACP channel.

4.6. Context Separation

The ACP is in a separate context from the normal data plane of the
device. This context includes the ACP channels IPv6 forwarding and
routing as well as any required higher layer ACP functions.

In classical network device platforms, a dedicated so called "Virtual
routing and forwarding instance" (VRF) is one logical implementation
option for the ACP. If possible by the platform SW architecture,
separation options that minimize shared components are preferred.
The context for the ACP needs to be established automatically during
bootstrap of a device and - as necessitated by the implementation
option be protected from being modified unintentional from data plane
configuration.

In addition this provides for security, because the ACP is not
reachable from the global routing table. Also, configuration errors
from the data plane setup do not affect the ACP.

4.7. Addressing inside the ACP

The channels explained above only establish communication between two
adjacent neighbors. In order for the communication to happen across
multiple hops, the autonomic control plane requires internal network
wide valid addresses and routing. Each autonomic node must create a
loopback interface with a network wide unique address inside the ACP
context mentioned in Section 4.6.

We suggest to create network wide Unique Local Addresses (ULA) in
accordance with [RFC4193] with the following algorithm:

- Prefix FC01::/8
- Global ID: a hash of the domain ID; this way all devices in the
  same domain have the same /48 prefix. Conversely, global ID from
different domains are unlikely to clash, such that two networks
can be merged, as long as the policy allows that merge. See also
Section 5 for a discussion on merging domains.
- Subnet ID and interface ID: These can be either derived
deterministically from the name of the device, or assigned at
registration time of the device.
Links inside the ACP only use link-local IPv6 addressing, such that each node only requires one routable loopback address.

4.8. 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 the global routing table 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 manual configuration is required.

All routing updates are automatically secured in transit as the channels of the autonomic control plane are by default secured.

The routing protocol inside the ACP should be light weight and highly scalable to ensure that the ACP does not become a limiting factor in network scalability. We suggest the use of RPL as one such protocol which is light weight and scales well for the control plane traffic.

4.9. Connecting a 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 through it. For this, an NMS host must have access to the ACP. By default, the ACP is a self-protecting overlay network, which only allows access to trusted systems. Therefore, a traditional NMS system does not have access to the ACP by default, just like any other external device.

The preferred way for an NMS host to connect to the ACP of a network is to enrol that NMS host as a domain device, such that it shares a domain certificate with the same trust anchor as the network devices. Then, the NMS host can automatically discover an adjacent network element, and join the ACP automatically, just like a network device would connect to a neighboring device. Alternatively, if there is no directly connected autonomic network element, a secure connection to a single remote network element can be established by configuration, authenticated using the domain certificates. There, the NMS host "enters" the ACP, from which point it can use the ACP to reach further nodes.
If the NMS host does not support autonomic negotiation of the ACP, then it can be brought into the ACP by configuration. On an adjacent autonomic node with ACP, the interface with the NMS host can be configured to be part of the ACP. In this case, the NMS host is with this interface entirely and exclusively inside the ACP. It would likely require a second interface for connections between the NMS host and administrators, or Internet based services. This mode of connecting an NMS host has security consequences: All systems and processes connected to this implicitly trusted interface have access to all autonomic nodes on the entire ACP, without further authentication. Thus, this connection must be physically controlled.

In both options, the NMS host must be routed in the ACP. This involves two parts: 1) the NMS host must point default to the AN device for all IPv6, or for the ULA prefix used inside the ACP, and 2) the prefix used between AN node and NMS host must be announced into the ACP, and distributed there.

The document "Autonomic Network Stable Connectivity" [I-D.eckert-anima-stable-connectivity] explains in more detail how the ACP can be integrated in a mixed NOC environment.

5. 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 devices.

- If an existing device gets revoked, it will automatically be denied access to the ACP as its domain certificate will be validated against a Certificate Revocation List during authentication. Since the revocation check is only done at the establishment of a new security association, existing ones are not automatically torn down. If an immediate disconnect is required, existing sessions to a freshly revoked device can be re-set.

The ACP can also sustain network partitions and mergers. Practically all ACP operations are link local, where a network partition has no impact. Devices authenticate each other using the domain certificates to establish the ACP locally. Addressing inside the ACP remains unchanged, and the routing protocol inside both parts of the ACP will lead to two working (although partitioned) ACPs.
There are few central dependencies: A certificate revocation list (CRL) may not be available during a network partition; a suitable policy to not immediately disconnect neighbors when no CRL is available can address this issue. Also, a registrar or Certificate Authority might not be available during a partition. This may delay renewal of certificates that are to expire in the future, and it may prevent the enrolment of new devices during the partition.

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

Merging two networks with different trust anchors requires the trust anchors to mutually trust each other (for example, by cross-signing). As long as the domain names are different, the addressing will not overlap (see Section 4.7).

6. Self-Protection Properties

As explained in Section 4, the ACP is based on channels being built between devices which have been previously authenticated based on their domain certificates. The channels themselves are protected using standard encryption technologies like DTLS or IPsec which provide additional authentication during channel establishment, data integrity and data confidentiality protection of data inside the ACP and in addition, provide replay protection.

An attacker will therefore not be able to join the ACP unless having a valid domain certificate, also packet injection and sniffing traffic will not be possible due to the security provided by the encryption protocol.

The remaining attack vector would be to attack the underlying AN protocols themselves, either via directed attacks or by denial-of-service attacks. However, as the ACP is built using link-local IPv6 address, remote attacks are impossible. 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.

7. The Administrator View

An ACP is self-forming, self-managing and self-protecting, therefore has minimal dependencies on the administrator of the network. Specifically, it cannot be configured, there is therefore no 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 devices, except a
possible on/off switch.

While configuration 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 aspect unless they explicitly desire to
do so.

Since an ACP is self-protecting, a device 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 4.9 for more details on how to
connect an NMS host into the ACP.

8. Security Considerations

An ACP is self-protecting and there is no need to apply configuration
to make it secure. Its security therefore does not depend on
configuration.

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.

- Security can be compromised by implementation errors (bugs), as in
  all products.

Fundamentally, security depends on correct operation, implementation
and architecture. Autonomic approaches such as the ACP largely
eliminate the dependency on correct operation; implementation and
architectural mistakes are still possible, as in all networking
technologies.
9. IANA Considerations

This document requests no action by IANA.

10. 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, Alex Clemm, Toerless Eckert, Yves Hertoghs, Bruno Klauser, Max Pritikin, Ravi Kumar Vadapalli.

Further input and suggestions were received from: Rene Struik, Brian Carpenter, Benoit Claise.

11. Change log [RFC Editor: Please remove]

11.1. Initial version

First version of this document:
[I-D.behringer-autonomic-control-plane]

11.2. version 00

Initial version of the anima document; only minor edits.

11.3. version 01

- Clarified that the ACP should be based on, and support only IPv6.
- Clarified in intro that ACP is for both, between devices, as well as for access from a central entity, such as an NMS.
- Added a section on how to connect an NMS system.
- Clarified the hop-by-hop crypto nature of the ACP.
- Added several references to GDNP as a candidate protocol.
- Added a discussion on network split and merge. Although, this should probably go into the certificate management story longer term.
11.4. version 02

Addresses (numerous) comments from Brian Carpenter. See mailing list for details. The most important changes are:

- Introduced a new section "overview", to ease the understanding of the approach.
- Merged the previous "problem statement" and "use case" sections into a mostly re-written "use cases" section, since they were overlapping.
- Clarified the relationship with draft-eckert-anima-stable-connectivity

12. References

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

[I-D.behringer-autonomic-control-plane]

[I-D.carpenter-anima-gdn-protocol]

[I-D.eckert-anima-stable-connectivity]

[I-D.irtf-nmrg-an-gap-analysis]
(I-D.irtf-nmrg-autonomic-network-definitions)
Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A.,
Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic
Networking - Definitions and Design Goals", draft-irtf-
nmrg-autonomic-network-definitions-05 (work in progress),
December 2014.

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

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally
Unique IDentifier (UUID) URN Namespace", RFC 4122, July
2005.


Authors’ Addresses

Michael H. Behringer
Cisco
Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco
Email: sbjarnas@cisco.com

Balaji BL
Cisco
Email: blbalaji@cisco.com

Toerless Eckert
Cisco
Email: eckert@cisco.com
A Reference Model for Autonomic Networking
draft-behringer-anima-reference-model-00

Abstract

This document describes a reference model for Autonomic Networking. The goal is to define how the various elements in an autonomic context work together, to describe their interfaces and relations.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 20, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

The document "Autonomic Networking - Definitions and Design Goals" [I-D.irtf-nmrg-autonomic-network-definitions] explains the fundamental concepts behind Autonomic Networking, and defines the relevant terms in this space. In section 5 it describes a high level reference model. This document defines this reference model with more detail, to allow for functional and protocol specifications to be developed in an architecturally consistent, non-overlapping manner.

As discussed in [I-D.irtf-nmrg-autonomic-network-definitions], the goal of this work is not to focus exclusively on fully autonomic nodes or networks. In reality, most networks will run with some autonomic functions, while the rest of the network is traditionally managed. This reference model allows for this hybrid approach.
2. The Network View

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

Autonomic entities include:

- Network elements: A network element can be a fully or partially autonomic node. It runs autonomic functions, and interacts with other autonomic nodes.

- Registrar: Security is a fundamental requirement in an autonomic network. For nodes and services to securely interact without the need to provision shared secrets, a trust infrastructure must be in place. The registrar is the trust anchor in an autonomic domain.

- MASA: The MASA is service for devices of a particular vendor. It can validate the identity of devices that are to be used in an autonomic domain, assert which device is owned by which domain, etc.

3. Entities in an Autonomic Network

This section describes all the elements in an autonomic network, their function, internal organisation and architecture. In the network view in Section 2, this section describes the "boxes". The following sections describes how those boxes interact, and the necessary means to do so (addressing, routing, etc).

3.1. The Network Element

This section describes an autonomic network element and its internal architecture. The reference model explained in [I-D.irtf-nmrg-autonomic-network-definitions] shows the sources of information that an autonomic service agent can leverage: Self-knowledge, network knowledge (through discovery), Intent, and feedback loops. Fundamentally, there are two levels inside an autonomic node: the level of autonomic service agents, which uses the functions of the autonomic networking infrastructure. Figure 1 illustrates this concept.
The Autonomic Networking Infrastructure (lower part of Figure 1) contains node specific data structures, for example trust information about itself and its peers, as well as a generic set of functions, independent of a particular usage. This infrastructure should be generic, and support a variety of Autonomic Service Agents (upper part of Figure 1). The Autonomic Control Plane is the summary of all interactions of the Autonomic Networking Infrastructure with other nodes and services.

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

3.2. The Registrar Element

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

tbc

4. Naming

Inside a domain, each autonomic device needs a domain specific identifier. [tbc]

5. Addressing

tbc

6. Trust Infrastructure

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

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

7. Autonomic Control Plane

This section describes how autonomic nodes interact. In the network view in Section 2 this section describes the "lines" and "arrows" between nodes. The summary of autonomic interactions forms the "Autonomic Control Plane". This control plane can be either implemented in the global routing table of a node, such as IGPs in today’s networks; or it can be provided as an overlay network, as described in [I-D.behringer-autonomic-control-plane]. This section describes the function of the autonomic control plane, independent of its implementation.

7.1. Discovery

Traditionally, most of the information a node requires is provided through configuration or northbound interfaces. An autonomic function should only minimally rely on such northbound interfaces, therefore it needs to discover resources in the network. This section describes various discovery functions in an autonomic network.
Discovering nodes and their properties: A core function to establish an autonomic domain is the discovery of autonomic nodes, primarily adjacent nodes. This may either leverage existing neighbour discovery mechanisms, or new mechanisms.

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

7.2. Negotiation and Synchronisation

Autonomic nodes must negotiate and/or synchronise parameters, etc. The document "A Generic Discovery and Negotiation Protocol for Autonomic Networking" [I-D.carpenter-anima-gdn-protocol] explains requirements for negotiation and synchronisation in an autonomic network, and a protocol for this purpose.

7.3. Intent Distribution

The distribution of intent is also a function of the Autonomic Control Plane. Various methods can be used to distribute intent across an autonomic domain.

7.4. Reporting

An autonomic network offers through the autonomic control plane the possibility to aggregate information inside the network, before sending it to the admin of the network. While this can be seen or implemented as a specific form of negotiation, the use case is different and therefore mentioned here explicitly.

7.5. Feedback Loops

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

7.6. Routing

All autonomic nodes in a domain must be able to communicate with each other, and with autonomic nodes outside their own domain. Therefore, an Autonomic Control Plane relies on a routing function.
8. Hybrid Approach with Non-Autonomic Functions

This section explains how autonomic functions can co-exist with non-autonomic functions, and how a potential overlap is managed. (tbc)

9. Security Considerations

9.1. Threat Analysis

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

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

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

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

This document requests no action by IANA.

11. Acknowledgements

tbc

12. Change log [RFC Editor: Please remove]

00: Initial version.

13. References

[I-D.behringer-autonomic-control-plane]

[I-D.carpenter-anima-gdn-protocol]

[I-D.irtf-nmrg-autonomic-network-definitions]

[I-D.pritikin-bootstrapping-keyinfrastructures]

Authors' Addresses

Michael H. Behringer (editor)
Cisco

Email: mbehring@cisco.com
A Generic Discovery and Negotiation Protocol for Autonomic Networking
draft-carpenter-anima-gdn-protocol-02

Abstract

This document establishes requirements for a protocol that enables intelligent devices to dynamically discover peer devices, to synchronize state with them, and to negotiate parameter settings mutually with them. The document then defines a general protocol for discovery, synchronization and negotiation, while the technical objectives for specific scenarios are to be described in separate documents. An Appendix briefly discusses existing protocols with comparable features.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 23, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction .................................................. 3
2. Requirement Analysis of Discovery, Synchronization and Negotiation ............................................ 4
   2.1. Requirements for Discovery .............................. 4
   2.2. Requirements for Synchronization and Negotiation Capability .............................................. 5
   2.3. Specific Technical Requirements .......................... 7
3. GDNP Protocol Overview ........................................... 8
   3.1. Terminology ................................................ 8
   3.2. High-Level Design Choices ................................ 10
   3.3. GDNP Protocol Basic Properties and Mechanisms .......... 13
       3.3.1. Discovery Mechanism and Procedures ................. 13
       3.3.2. Certificate-based Security Mechanism ................ 15
       3.3.3. Negotiation Procedures ............................. 18
       3.3.4. Synchronization Procedure .......................... 19
   3.4. GDNP Constants ............................................ 20
   3.5. Device Identifier and Certificate Tag .................... 20
   3.6. Session Identifier (Session ID) .......................... 21
   3.7. GDNP Messages .............................................. 21
       3.7.1. GDNP Message Format ................................ 21
       3.7.2. Discovery Message .................................... 22
       3.7.3. Response Message ..................................... 23
       3.7.4. Request Message ....................................... 23
       3.7.5. Negotiation Message .................................. 24
       3.7.6. Negotiation-ending Message ........................... 24
       3.7.7. Confirm-waiting Message ............................... 24
   3.8. GDNP General Options ...................................... 25
       3.8.1. Format of GDNP Options ................................. 25
       3.8.2. Divert Option .......................................... 25
       3.8.3. Accept Option .......................................... 26
       3.8.4. Decline Option ......................................... 26
       3.8.5. Waiting Time Option ................................... 27
       3.8.6. Certificate Option ..................................... 28
       3.8.7. Signature Option ........................................ 28
       3.8.8. Locator Options ........................................ 29
   3.9. Objective Options .......................................... 31
       3.9.1. Format of Objective Options ............................ 31
       3.9.2. General Considerations for Objective Options ........ 32
       3.9.3. Organizing of Objective Options ........................ 32
       3.9.4. Vendor Specific Objective Options ...................... 33
       3.9.5. Experimental Objective Options ........................ 34
1. Introduction

The success of the Internet has made IP-based networks bigger and more complicated. Large-scale ISP and enterprise networks have become more and more problematic for human based management. Also, operational costs are growing quickly. Consequently, there are increased requirements for autonomic behavior in the networks. General aspects of autonomic networks are discussed in [I-D.irtf-nmrg-autonomic-network-definitions] and [I-D.irtf-nmrg-an-gap-analysis]. In order to fulfill autonomy, devices that embody autonomic service agents need to be able to discover each other, to synchronize state with each other, and to negotiate parameters and resources directly with each other. There is no restriction on the type of parameters and resources concerned, which include very basic information needed for addressing and routing, as well as anything else that might be configured in a conventional network.

Following this Introduction, Section 2 describes the requirements for network device discovery, synchronization and negotiation. Negotiation is an iterative process, requiring multiple message exchanges forming a closed loop between the negotiating devices. State synchronization, when needed, can be regarded as a special case of negotiation, without iteration. Section 3.2 describes a behavior model for a protocol intended to support discovery, synchronization and negotiation. The design of Generic Discovery and Negotiation Protocol (GDNP) in Section 3 of this document is mainly based on this behavior model. The relevant capabilities of various existing protocols are reviewed in Appendix A.

The proposed discovery mechanism is oriented towards synchronization and negotiation objectives. It is based on a neighbor discovery process, but also supports diversion to off-link peers. Although many negotiations will occur between horizontally distributed peers, many target scenarios are hierarchical networks, which is the predominant structure of current large-scale networks. However, when a device starts up with no pre-configuration, it has no knowledge of
a hierarchical superior. The protocol itself is capable of being used in a small and/or flat network structure such as a small office or home network as well as a professionally managed network. Therefore, the discovery mechanism needs to be able to allow a device to bootstrap itself without making any prior assumptions about network structure.

Because GDNP can be used to perform a decision process among distributed devices or between networks, it adopts a tight certificate-based security mechanism, which needs a Public Key Infrastructure (PKI) [RFC5280] system. The PKI may be managed by an operator or be autonomic, as discussed in [I-D.pritikin-anima-bootstrapping-keyinfra].

It is understood that in realistic deployments, not all devices will support GDNP. It is expected that some autonomic service agents will manage a group of non-autonomic nodes, and that other non-autonomic nodes will be managed traditionally. Such mixed scenarios are not discussed in this specification.

2. Requirement Analysis of Discovery, Synchronization and Negotiation

This section discusses the requirements for discovery, negotiation and synchronization capabilities.

2.1. Requirements for Discovery

In an autonomic network we must assume that when a device starts up it has no information about any peer devices, the network structure, or what specific role it must play. In some cases, when a new application session starts up within a device, the device may again lack information about relevant peer devices. It might be necessary to set up resources on multiple other devices, coordinated and matched to each other so that there is no wasted resource. Security settings might also need updating to allow for the new device or user. Therefore a basic requirement is that there must be a mechanism by which a device can separately discover peer devices for each of the technical objectives that it needs to manage. Some objectives may only be significant on the local link, but others may be significant across the routed network and require off-link operations. Thus, the relevant peer devices might be immediate neighbors on the same layer 2 link or they might be more distant and only accessible via layer 3. The mechanism must therefore support both on-link discovery and off-link discovery of peers that support specific technical objectives.

The relevant peer devices may be different for different technical objectives. Therefore discovery needs to be repeated as often as
necessary to find peers capable of acting as counterparts for each objective that a discovery initiator needs to handle. In many scenarios, the discovery process may be followed by a synchronization or negotiation process. Therefore, a discovery objective may be associated with one or more synchronization or negotiation objectives.

When a device first starts up, it has no knowledge of the network structure. Therefore the discovery process must be able to support any network scenario, assuming only that the device concerned is bootstrapped from factory condition.

In some networks, as mentioned above, there will be some hierarchical structure, at least for certain synchronization or negotiation objectives. A special case of discovery is that each device must be able to discover its hierarchical superior for each such objective that it is capable of handling. This is part of the more general requirement to discover off-link devices.

During initialisation, a device must be able to establish mutual trust with the rest of the network and join the PKI. Although this must inevitably start with a discovery action, it is a special case precisely because trust is not yet established. This topic is the subject of [I-D.pritikin-anima-bootstrapping-keyinfra]. In addition, depending on the type of network involved, discovery of other central functions might be needed, such as the Network Operations Center (NOC) [I-D.eckert-anima-stable-connectivity].

2.2. Requirements for Synchronization and Negotiation Capability

We start by considering routing protocols, the closest approximation to autonomic networking in widespread use. Routing protocols use a largely autonomic model based on distributed devices that communicate repeatedly with each other. However, routing is mainly based on one-way information synchronization (in either direction), rather than on bi-directional negotiation. The focus is reachability, so current routing protocols only consider simple link status, i.e., up or down. More information, such as latency, congestion, capacity, and particularly unused capacity, would be helpful to get better path selection and utilization rate. Also, autonomic networks need to be able to manage many more dimensions, such as security settings, power saving, load balancing, etc. A basic requirement for the protocol is therefore the ability to represent, discover, synchronize and negotiate almost any kind of network parameter.

Human intervention in complex situations is costly and error-prone. Therefore, synchronization or negotiation of parameters without human intervention is desirable whenever the coordination of multiple
devices can improve overall network performance. It follows that a
requirement for the protocol is to be capable of running in any
device that would otherwise need human intervention.

Human intervention in large networks is often replaced by use of a
top-down network management system (NMS). It therefore follows that
a requirement for the protocol is to be capable of running in any
device that would otherwise be managed by an NMS, and that it can co-
exist with an NMS.

Since the goal is to minimize human intervention, it is necessary
that the network can in effect "think ahead" before changing its
parameters. In other words there must be a possibility of
forecasting the effect of a change by a "dry run" mechanism before
actually installing the change. This will be an application of the
protocol rather than a feature of the protocol itself.

Status information and traffic metrics need to be shared between
nodes for dynamic adjustment of resources and for monitoring
purposes. While this might be achieved by existing protocols when
they are available, the new protocol needs to be able to support
parameter exchange, including mutual synchronization, even when no
negotiation as such is required.

Recovery from faults and identification of faulty devices should be
as automatic as possible. However, the protocol’s role is limited to
the ability to handle discovery, synchronization and negotiation at
any time, in case an autonomic service agent detects an anomaly such
as a negotiation counterpart failing.

Management logging, monitoring, alerts and tools for intervention are
required. However, these can only be features of individual
autonomic service agents. Another document
[I-D.eckert-anima-stable-connectivity] discusses how such agents may
be linked into conventional OAM systems via an Autonomic Control
Plane [I-D.behringer-anima-autonomic-control-plane].

The protocol needs to be able to deal with a wide variety of
technical objectives, covering any type of network parameter.
Therefore the protocol will need either an explicit information model
describing its messages, or at least a flexible and extensible
message format. One design consideration is whether to adopt an
existing information model or to design a new one. Another
consideration is whether it should be able to carry some or all of
the message formats used by existing configuration protocols.
2.3. Specific Technical Requirements

To be a generic platform, the protocol payload format should be independent of the transport protocol or IP version. In particular, it should be able to run over IPv6 or IPv4. However, some functions, such as multicasting or broadcasting on a link, might need to be IP version dependent. In case of doubt, IPv6 should be preferred.

The protocol must be able to access off-link counterparts via routable addresses, i.e., must not be restricted to link-local operation.

The negotiation process must be guaranteed to terminate (with success or failure) and if necessary it must contain tie-breaking rules for each technical objective that requires them. While this must be defined specifically for each use case, the protocol should have some general mechanisms in support of loop and deadlock prevention.

Dependencies: In order to decide a configuration on a given device, the device may need information from neighbors. This can be established through the negotiation procedure, or through synchronization if that is sufficient. However, a given item in a neighbor may depend on other information from its own neighbors, which may need another negotiation or synchronization procedure to obtain or decide. Therefore, there are potential dependencies among negotiation or synchronization procedures. Thus, there need to be clear boundaries and convergence mechanisms for these negotiation dependencies. Also some mechanisms are needed to avoid loop dependencies.

Policy constraints: There must be provision for general policy intent rules to be applied by all devices in the network (e.g., security rules, prefix length, resource sharing rules). However, policy intent distribution might not use the negotiation protocol itself.

Management monitoring, alerts and intervention: Devices should be able to report to a monitoring system. Some events must be able to generate operator alerts and some provision for emergency intervention must be possible (e.g. to freeze synchronization or negotiation in a mis-behaving device). These features may not use the negotiation protocol itself.

The protocol needs to be fully secure against forged messages and man-in-the-middle attacks, and as secure as reasonably possible against denial of service attacks. It needs to be capable of encryption in order to resist unwanted monitoring, although this capability may not be required in all deployments.
3. GDNP Protocol Overview

3.1. Terminology

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

The following terms are used throughout this document:

- Discovery: a process by which a device discovers peer devices according to a specific discovery objective. The discovery results may be different according to the different discovery objectives. The discovered peer devices may later be used as negotiation counterparts or as sources of synchronization data.

- Negotiation: a process by which two (or more) devices interact iteratively to agree on parameter settings that best satisfy the objectives of one or more devices.

- State Synchronization: a process by which two (or more) devices interact to agree on the current state of parameter values stored in each device. This is a special case of negotiation in which information is sent but the devices do not request their peers to change parameter settings. All other definitions apply to both negotiation and synchronization.

- Objective: An objective in GDNP is a configurable state of some kind, which occurs in three contexts: Discovery, Negotiation and Synchronization. In the protocol, an objective is represented by an identifier (actually a GDNP option number) and if relevant a value. Normally, a given objective will occur during discovery and negotiation, or during discovery and synchronization, but not in all three contexts.

- One device may support multiple independent objectives.

- The parameter described by a given objective is naturally based on a specific service or function or action. It may in principle be anything that can be set to a specific logical, numerical or string value, or a more complex data structure, by a network node. That node is generally expected to be an autonomic service agent which may itself manage other nodes.
* Discovery Objective: if a node needs to synchronize or negotiate a specific objective but does not know a peer that supports this objective, it starts a discovery process. The objective is called a Discovery Objective during this process.

* Synchronization Objective: an objective whose specific technical content needs to be synchronized among two or more devices.

* Negotiation Objective: an objective whose specific technical content needs to be decided in coordination with another network device.

  o Discovery Initiator: a device that spontaneously starts discovery by sending a discovery message referring to a specific discovery objective.

  o Discovery Responder: a peer device which responds to the discovery objective initiated by the discovery initiator.

  o Synchronization Initiator: a device that spontaneously starts synchronization by sending a request message referring to a specific synchronization objective.

  o Synchronization Responder: a peer device which responds with the value of a synchronization objective.

  o Negotiation Initiator: a device that spontaneously starts negotiation by sending a request message referring to a specific negotiation objective.

  o Negotiation Counterpart: a peer device with which the Negotiation Initiator negotiates a specific negotiation objective.

  o Device Identifier: a public key, which identifies the device in GDNP messages. It is assumed that its associated private key is maintained in the device only.

  o Device Certificate: A certificate for a single device, also the identifier of the device, further described in Section 3.5.

  o Device Certificate Tag: a tag, which is bound to the device identifier. It is used to present a Device Certificate in short form.
3.2. High-Level Design Choices

This section describes a behavior model and some considerations for designing a generic discovery, synchronization and negotiation protocol, which can act as a platform for different technical objectives.

NOTE: This protocol is described here in a stand-alone fashion as a proof of concept. An elementary version has been prototyped by Huawei and the Beijing University of Posts and Telecommunications. However, this is not yet a definitive proposal for IETF adoption. In particular, adaptation and extension of one of the protocols discussed in Appendix A might be an option. Also, the security model outlined below would in practice be part of a general security mechanism in an autonomic control plane [I-D.behringer-anima-autonomic-control-plane]. This whole specification is subject to change as a result.

- A generic platform

  The protocol is designed as a generic platform, which is independent from the synchronization or negotiation contents. It takes care of the general intercommunication between counterparts. The technical contents will vary according to the various synchronization or negotiation objectives and the different pairs of counterparts.

- Security infrastructure and trust relationship

  Because this negotiation protocol may directly cause changes to device configurations and bring significant impacts to a running network, this protocol is based on a restrictive security infrastructure, allowing it to be trusted and monitored so that every device in this negotiation system behaves well and remains well protected.

  On the other hand, a limited negotiation model might be deployed based on a limited trust relationship. For example, between two administrative domains, devices might also exchange limited information and negotiate some particular configurations based on a limited conventional or contractual trust relationship.

- Discovery, synchronization and negotiation designed together

  The discovery method and the synchronization and negotiation methods are designed in the same way and can be combined when this
is useful. These processes can also be performed independently when appropriate.

- **A uniform pattern for technical contents**

  The synchronization and negotiation contents are defined according to a uniform pattern. They could be carried either in simple TLV (Type, Length and Value) format or in payloads described by a flexible language. The initial protocol design uses the TLV approach. The format is extensible for unknown future requirements.

- **A conservative model for synchronization**

  Synchronization across a number of nodes is not a new problem and the Trickle model that is already known to be effective and efficient is suggested.

- **A simple initiator/responder model for negotiation**

  Multi-party negotiations are too complicated to be modeled and there might be too many dependencies among the parties to converge efficiently. A simple initiator/responder model is more feasible and can complete multiple-party negotiations by indirect steps.

- **Organizing of synchronization or negotiation content**

  Naturally, the technical content will be organized according to the relevant function or service. The content from different functions or services is kept independent from each other. They are not combined into a single option or single session because these contents may be negotiated or synchronized with different counterparts or may be different in response time.

- **Self aware network device**

  Every network device will be pre-loaded with various functions and be aware of its own capabilities, typically decided by the hardware, firmware or pre-installed software. Its exact role may depend on the surrounding network behaviors, which may include forwarding behaviors, aggregation properties, topology location, bandwidth, tunnel or translation properties, etc. The surrounding topology will depend on the network planning. Following an
initial discovery phase, the device properties and those of its neighbors are the foundation of the synchronization or negotiation behavior of a specific device. A device has no pre-configuration for the particular network in which it is installed.

- Requests and responses in negotiation procedures

The initiator can negotiate with its relevant negotiation counterpart devices, which may be different according to the specific negotiation objective. It can request relevant information from the negotiation counterpart so that it can decide its local configuration to give the most coordinated performance. It can request the negotiation counterpart to make a matching configuration in order to set up a successful communication with it. It can request certain simulation or forecast results by sending some dry run conditions.

Beyond the traditional yes/no answer, the responder can reply with a suggested alternative if its answer is ‘no’. This would start a bi-directional negotiation ending in a compromise between the two devices.

- Convergence of negotiation procedures

To enable convergence, when a responder makes a suggestion of a changed condition in a negative reply, it should be as close as possible to the original request or previous suggestion. The suggested value of the third or later negotiation steps should be chosen between the suggested values from the last two negotiation steps. In any case there must be a mechanism to guarantee convergence (or failure) in a small number of steps, such as a timeout or maximum number of iterations.

* End of negotiation

A limited number of rounds, for example three, or a timeout, is needed on each device for each negotiation objective. It may be an implementation choice, a pre-configurable parameter, or a network-wide policy intent. These choices might vary between different types of autonomic service agent. Therefore, the definition of each negotiation objective MUST clearly specify this, so that the negotiation can always be terminated properly.
Failed negotiation

There must be a well-defined procedure for concluding that a negotiation cannot succeed, and if so deciding what happens next (deadlock resolution, tie-breaking, or revert to best-effort service). Again, this MUST be specified for individual negotiation objectives, as an implementation choice, a pre-configurable parameter, or a network-wide policy intent.

3.3. GDNP Protocol Basic Properties and Mechanisms

3.3.1. Discovery Mechanism and Procedures

- Separated discovery and negotiation mechanisms

Although discovery and negotiation or synchronization are defined together in the GDNP, they are separated mechanisms. The discovery process could run independently from the negotiation or synchronization process. Upon receiving a discovery (Section 3.7.2) or request (Section 3.7.4) message, the recipient device should return a message in which it either indicates itself as a discovery responder or diverts the initiator towards another more suitable device.

The discovery action will normally be followed by a negotiation or synchronization action. The discovery results could be utilized by the negotiation protocol to decide which device the initiator will negotiate with.

- Discovery Procedures

Discovery starts as an on-link operation. The Divert option can tell the discovery initiator to contact an off-link discovery objective device. Every DISCOVERY message is sent by a discovery initiator via UDP to the ALL_GDNP_NEIGHBOR multicast address (Section 3.4). Every network device that supports the GDNP always listens to a well-known UDP port to capture the discovery messages.

If the neighbor device supports the requested discovery objective, it MAY respond with a Response message (Section 3.7.3) with locator option(s). Otherwise, if the neighbor device has cached information about a device that supports the requested discovery objective (usually because it discovered the same objective before), it SHOULD respond with a Response message with a Divert option pointing to the appropriate Discovery Responder.
If no discovery response is received within a reasonable timeout (default GDNP_DEF_TIMEOUT milliseconds, Section 3.4), the DISCOVERY message MAY be repeated, with a newly generated Session ID (Section 3.6). An exponential backoff MAY be used for subsequent repetitions.

After a GDNP device successfully discovers a Discovery Responder supporting a specific objective, it MUST cache this information. This cache record MAY be used for future negotiation or synchronization, and SHOULD be passed on when appropriate as a Divert option to another Discovery Initiator. The cache lifetime is an implementation choice.

If multiple Discovery Responders are found for the same objective, they SHOULD all be cached, unless this creates a resource shortage. The method of choosing between multiple responders is an implementation choice.

A GDNP device with multiple link-layer interfaces (typically a router) MUST support discovery on all interfaces. If it receives a DISCOVERY message on a given interface for a specific objective that it does not support and for which it has not previously discovered a Discovery Responder, it MUST relay the query by re-issuing the same DISCOVERY message on its other interfaces. However, it SHOULD limit the total rate at which it relays discovery messages to a reasonable value. It MUST cache the Session ID value of each relayed discovery message and, to prevent loops, MUST NOT relay a DISCOVERY message which carries such a cached Session ID.

This relayed discovery mechanism, with caching of the results, should be sufficient to support most network bootstrapping scenarios.

- A complete discovery process will start with multicast on the local link; a neighbor might divert it to an off-link destination, which could be a default higher-level gateway in a hierarchical network. Then discovery would continue with a unicast to that gateway; if that gateway is still not the right counterpart, it should divert to another device, which is in principle closer to the right counterpart. Finally the right counterpart responds to start the negotiation or synchronization process.

- Rapid Mode (Discovery/Negotiation binding)

  A Discovery message MAY include one or more Negotiation Objective option(s). This allows a rapid mode of negotiation
described in Section 3.3.3. A similar mechanism is defined for synchronization.

3.3.2. Certificate-based Security Mechanism

A certificate-based security mechanism provides security properties for GDNP:

- the identity of a GDNP message sender can be verified by a recipient.
- the integrity of a GDNP message can be checked by the recipient of the message.
- anti-replay protection can be assured by the GDNP message recipient.

The authority of the GDNP message sender depends on a Public Key Infrastructure (PKI) system with a Certification Authority (CA), which should normally be run by the network operator. In the case of a network with no operator, such as a small office or home network, the PKI itself needs to be established by an autonomic process, which is out of scope for this specification.

A Request message MUST carry a Certificate option, defined in Section 3.8.6. The first Negotiation Message, responding to a Request message, SHOULD also carry a Certificate option. Using these messages, recipients build their certificate stores, indexed by the Device Certificate Tags included in every GDNP message. This process is described in more detail below.

Every message MUST carry a signature option (Section 3.8.7).

For now, the authors do not think packet size is a problem. In this GDNP specification, there SHOULD NOT be multiple certificates in a single message. The current most used public keys are 1024/2048 bits; some may reach 4096. With overhead included, a single certificate is less than 500 bytes. Messages are expected to be far shorter than the normal packet MTU within a modern network.

3.3.2.1. Support for algorithm agility

Hash functions are used to provide message integrity checks. In order to provide a means of addressing problems that may emerge in the future with existing hash algorithms, as recommended in [RFC4270], a mechanism for negotiating the use of more secure hashes in the future is provided.
In addition to hash algorithm agility, a mechanism for signature algorithm agility is also provided.

The support for algorithm agility in this document is mainly a unilateral notification mechanism from sender to recipient. If the recipient does not support the algorithm used by the sender, it cannot authenticate the message. Senders in a single administrative domain are not required to upgrade to a new algorithm simultaneously.

So far, the algorithm agility is supported by one-way notification, rather than negotiation mode. As defined in Section 3.8.7, the sender notifies the recipient what hash/signature algorithms it uses. If the responder doesn’t know a new algorithm used by the sender, the negotiation request would fail. In order to establish a negotiation session, the sender MAY fall back to an older, less preferred algorithm. Certificates and network policy intent SHOULD limit the choice of algorithms.

3.3.2.2. Message validation on reception

When receiving a GDNP message, a recipient MUST discard the GDNP message if the Signature option is absent, or the Certificate option is in a Request Message.

For the Request message and the Response message with a Certification Option, the recipient MUST first check the authority of this sender following the rules defined in [RFC5280]. After successful authority validation, an implementation MUST add the sender’s certification into the local trust certificate record indexed by the associated Device Certificate Tag (Section 3.5).

The recipient MUST now authenticate the sender by verifying the Signature and checking a timestamp, as specified in Section 3.3.2.3. The order of two procedures is left as an implementation decision. It is RECOMMENDED to check timestamp first, because signature verification is much more computationally expensive.

The signature field verification MUST show that the signature has been calculated as specified in Section 3.8.7. The public key used for signature validation is obtained from the certificate either carried by the message or found from a local trust certificate record by searching the message-carried Device Certificate Tag.

Only the messages that get through both the signature verifications and timestamp check are accepted and continue to be handled for their contained GDNP options. Messages that do not pass the above tests MUST be discarded as insecure messages.
3.3.2.3. TimeStamp checking

Recipients SHOULD be configured with an allowed timestamp Delta value, a "fuzz factor" for comparisons, and an allowed clock drift parameter. The recommended default value for the allowed Delta is 300 seconds (5 minutes); for fuzz factor 1 second; and for clock drift, 0.01 second.

The timestamp is defined in the Signature Option, Section 3.8.7. To facilitate timestamp checking, each recipient SHOULD store the following information for each sender:

- The receive time of the last received and accepted GDNP message. This is called RDlast.
- The time stamp in the last received and accepted GDNP message. This is called TSlast.

An accepted GDNP message is any successfully verified (for both timestamp check and signature verification) GDNP message from the given peer. It initiates the update of the above variables. Recipients MUST then check the Timestamp field as follows:

- When a message is received from a new peer (i.e., one that is not stored in the cache), the received timestamp, TSnew, is checked, and the message is accepted if the timestamp is recent enough to the reception time of the packet, RDnew:
  
  \[-\Delta < (RDnew - TSnew) < +\Delta\]

  The RDnew and TSnew values SHOULD be stored in the cache as RDlast and TSlast.

- When a message is received from a known peer (i.e., one that already has an entry in the cache), the timestamp is checked against the previously received GDNP message:
  
  \[TSnew + fuzz > TSlast + (RDnew - RDlast) \times (1 - \text{drift}) - fuzz\]

  If this inequality does not hold, the recipient SHOULD silently discard the message. If, on the other hand, the inequality holds, the recipient SHOULD process the message.

  Moreover, if the above inequality holds and TSnew > TSlast, the recipient SHOULD update RDlast and TSlast. Otherwise, the recipient MUST NOT update RDlast or TSlast.
An implementation MAY use some mechanism such as a timestamp cache to strengthen resistance to replay attacks. When there is a very large number of nodes on the same link, or when a cache filling attack is in progress, it is possible that the cache holding the most recent timestamp per sender will become full. In this case, the node MUST remove some entries from the cache or refuse some new requested entries. The specific policy as to which entries are preferred over others is left as an implementation decision.

3.3.3. Negotiation Procedures

A negotiation initiator sends a negotiation request to a counterpart device, including a specific negotiation objective. It may request the negotiation counterpart to make a specific configuration. Alternatively, it may request a certain simulation or forecast result by sending a dry run configuration. The details, including the distinction between dry run and an actual configuration change, will be defined separately for each type of negotiation objective.

If the counterpart can immediately apply the requested configuration, it will give an immediate positive (accept) answer. This will end the negotiation phase immediately. Otherwise, it will negotiate. It will reply with a proposed alternative configuration that it can apply (typically, a configuration that uses fewer resources than requested by the negotiation initiator). This will start a bi-directional negotiation to reach a compromise between the two network devices.

The negotiation procedure is ended when one of the negotiation peers sends a Negotiation Ending message, which contains an accept or decline option and does not need a response from the negotiation peer. Negotiation may also end in failure (equivalent to a decline) if a timeout is exceeded or a loop count is exceeded.

A negotiation procedure concerns one objective and one counterpart. Both the initiator and the counterpart may take part in simultaneous negotiations with various other devices, or in simultaneous negotiations about different objectives. Thus, GDNP is expected to be used in a multi-threaded mode. Certain negotiation objectives may have restrictions on multi-threading, for example to avoid over-allocating resources.

Rapid Mode (Discovery/Negotiation linkage)

A Discovery message MAY include a Negotiation Objective option. In this case the Discovery message also acts as a Request message to indicate to the Discovery Responder that it could directly reply to the Discovery Initiator with a Negotiation message for
rapid processing, if it could act as the corresponding negotiation counterpart. However, the indication is only advisory not prescriptive.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. This rapid negotiation function SHOULD be configured off by default and MAY be configured on or off by policy intent.

### 3.3.4. Synchronization Procedure

A synchronization initiator sends a synchronization request to a counterpart device, including a specific synchronization objective. The counterpart responds with a Response message containing the current value of the requested synchronization objective. No further messages are needed. If no Response message is received, the synchronization request MAY be repeated after a suitable timeout.

In the case just described, the message exchange is unicast and concerns only one synchronization objective. In the following two cases, multiple synchronization objectives may be combined.

A synchronization responder MAY send an unsolicited Response message containing one or more Synchronization Objective option(s), if and only if the specification of those objectives permits it. This MAY be sent as a multicast message to the ALL_GDNP_NEIGHBOR multicast address (Section 3.4). In this case a suitable mechanism is needed to avoid excessive multicast traffic. This mechanism MUST be defined as part of the specification of the synchronization objective(s) concerned. It might be a simple rate limit or a more complex mechanism such as the Trickle algorithm [RFC6206].

**Rapid Mode (Discovery/Synchronization linkage)**

A Discovery message MAY include one or more Synchronization Objective option(s). In this case the Discovery message also acts as a Request message to indicate to the Discovery Responder that it could directly reply to the Discovery Initiator with a Response message with synchronization data for rapid processing, if the discovery target supports the corresponding synchronization objective. However, the indication is only advisory not prescriptive.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. This rapid synchronization function SHOULD be configured off by default and MAY be configured on or off by policy intent.
3.4. GDNP Constants

- **ALL_GDNP_NEIGHBOR (TBD1)**
  A link-local scope multicast address used by a GDNP-enabled device to discover GDNP-enabled neighbor (i.e., on-link) devices. All devices that support GDNP are members of this multicast group.

  * IPv6 multicast address: TBD1
  * IPv4 multicast address: TBD2

- **GDNP Listen Port (TBD3)**
  A UDP and TCP port that every GDNP-enabled network device always listens to.

- **GDNP_DEF_TIMEOUT (60000 milliseconds)**
  The default timeout used to determine that a discovery or negotiation has failed to complete.

- **GDNP_DEF_LOOPCT (6)**
  The default loop count used to determine that a negotiation has failed to complete.

3.5. Device Identifier and Certificate Tag

A GDNP-enabled Device MUST generate a stable public/private key pair before it participates in GDNP. There MUST NOT be any way of accessing the private key via the network or an operator interface. The device then uses the public key as its identifier, which is cryptographic in nature. It is a GDNP unique identifier for a GDNP participant.

It then gets a certificate for this public key, signed by a Certificate Authority that is trusted by other network devices. The Certificate Authority SHOULD be managed within the local administrative domain, to avoid needing to trust a third party. The signed certificate would be used for authentication of the message sender. In a managed network, this certification process could be performed at a central location before the device is physically installed at its intended location. In an unmanaged network, this process must be autonomic, including the bootstrap phase.

A 128-bit Device Certificate Tag, which is generated by taking a cryptographic hash over the device certificate, is a short...
presentation for GDNP messages. It is the index key to find the
device certificate in a recipient’s local trusted certificate record.

The tag value is formed by taking a SHA-1 hash algorithm [RFC3174]
over the corresponding device certificate and taking the leftmost 128
bits of the hash result.

3.6. Session Identifier (Session ID)

A 24-bit opaque value used to distinguish multiple sessions between
the same two devices. A new Session ID MUST be generated for every
new Discovery or Request message, and for every unsolicited Response
message. All follow-up messages in the same discovery,
synchronization or negotiation procedure, which is initiated by the
request message, MUST carry the same Session ID.

The Session ID SHOULD have a very low collision rate locally. It is
RECOMMENDED to be generated by a pseudo-random algorithm using a seed
which is unlikely to be used by any other device in the same network
[RFC4086].

3.7. GDNP Messages

This document defines the following GDNP message format and types.
Message types not listed here are reserved for future use. The
numeric encoding for each message type is shown in parentheses.

3.7.1. GDNP Message Format

All GDNP messages share an identical fixed format header and a
variable format area for options. Every Message carries the Device
Certificate Tag of its sender and a Session ID. Options are
presented serially in the options field, with no padding between the
options. Options are byte-aligned.

The following diagram illustrates the format of GDNP messages:
MESSAGE_TYPE:  Identifies the GDNP message type. 8-bit.

Session ID:  Identifies this negotiation session, as defined in Section 3.6. 24-bit.

Device Certificate Tag:  Represents the Device Certificate, which identifies the negotiation devices, as defined in Section 3.5. The Device Certificate Tag is 128 bit, also defined in Section 3.5. It is used as index key to find the device certificate.

Options:  GDNP Options carried in this message. Options are defined starting at Section 3.8.

3.7.2. Discovery Message

DISCOVERY (MESSAGE_TYPE = 1):

A discovery initiator sends a DISCOVERY message to initiate a discovery process.

The discovery initiator sends the DISCOVERY messages to the link-local ALL_GDNP_NEIGHBOR multicast address for discovery, and stores the discovery results (including responding discovery objectives and corresponding unicast addresses or FQDNs).

A DISCOVERY message MUST include exactly one of the following:

- a discovery objective option (Section 3.9.1).
- a negotiation objective option (Section 3.9.1) to indicate to the discovery target that it MAY directly reply to the discovery initiator with a NEGOTIATION message for rapid processing, if it could act as the corresponding negotiation counterpart. The
sender of such a DISCOVERY message MUST initialize a negotiation
timer and loop count in the same way as a REQUEST message
(Section 3.7.4).

- one or more synchronization objective options (Section 3.9.1) to
  indicate to the discovery target that it MAY directly reply to the
discovery initiator with a RESPONSE message for rapid processing,
if it could act as the corresponding synchronization counterpart.

3.7.3. Response Message

RESPONSE (MESSAGE_TYPE = 2):

A node which receives a DISCOVERY message sends a Response message to
respond to a discovery. It MUST contain the same Session ID as the
DISCOVERY message. It MAY include a copy of the discovery objective
from the DISCOVERY message.

If the responding node supports the discovery objective of the
discovery, it MUST include at least one kind of locator option
(Section 3.8.8) to indicate its own location. A combination of
multiple kinds of locator options (e.g. IP address option + FQDN
option) is also valid.

If the responding node itself does not support the discovery
objective, but it knows the locator of the discovery objective, then
it SHOULD respond to the discovery message with a divert option
(Section 3.8.2) embedding a locator option or a combination of
multiple kinds of locator options which indicate the locator(s) of
the discovery objective.

A node which receives a synchronization request sends a Response
message with the synchronization data. A node MAY send an
unsolicited Response Message with synchronization data and this MAY
be sent to the link-local ALL_GDNP_NEIGHBOR multicast address, in
accordance with the rules in Section 3.3.4.

If the response contains synchronization data, this will be in the
form of GDNP Option(s) for the specific synchronization objective(s).

3.7.4. Request Message

REQUEST (MESSAGE_TYPE = 3):

A negotiation or synchronization requesting node sends the REQUEST
message to the unicast address (directly stored or resolved from the
FQDN) of the negotiation or synchronization counterpart (selected
from the discovery results).
A request message MUST include the relevant objective option, with the requested value in the case of negotiation.

When an initiator sends a REQUEST message, it MUST initialize a negotiation timer for the new negotiation thread with the value GDNP_DEF_TIMEOUT milliseconds. Unless this timeout is modified by a CONFIRM-WAITING message (Section 3.7.7), the initiator will consider that the negotiation has failed when the timer expires.

When an initiator sends a REQUEST message, it MUST initialize the loop count of the objective option with a value defined in the specification of the option or, if no such value is specified, with GDNP_DEF_LOOPCT.

3.7.5. Negotiation Message

NEGOTIATION (MESSAGE_TYPE = 4):

A negotiation counterpart sends a NEGOTIATION message in response to a REQUEST message, a NEGOTIATION message, or a DISCOVERY message in Rapid Mode. A negotiation process MAY include multiple steps.

The NEGOTIATION message MUST include the relevant Negotiation Objective option, with its value updated according to progress in the negotiation. The sender MUST decrement the loop count by 1. If the loop count becomes zero both parties will consider that the negotiation has failed.

3.7.6. Negotiation-ending Message

NEGOTIATION-ENDING (MESSAGE_TYPE = 5):

A negotiation counterpart sends an NEGOTIATION-ENDING message to close the negotiation. It MUST contain one, but only one of accept/decline option, defined in Section 3.8.3 and Section 3.8.4. It could be sent either by the requesting node or the responding node.

3.7.7. Confirm-waiting Message

CONFIRM-WAITING (MESSAGE_TYPE = 6):

A responding node sends a CONFIRM-WAITING message to indicate the requesting node to wait for a further negotiation response. It might be that the local process needs more time or that the negotiation depends on another triggered negotiation. This message MUST NOT include any other options than the Waiting Time Option (Section 3.8.5).
3.8. GDNP General Options

This section defines the GDNP general option for the negotiation and synchronization protocol signalling. Option types 10–63 are reserved for GDNP general options defined in the future.

3.8.1. Format of GDNP Options

```
          0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----------------------------------------------+
<table>
<thead>
<tr>
<th>option-code</th>
<th>option-len</th>
</tr>
</thead>
<tbody>
<tr>
<td>option-data</td>
<td></td>
</tr>
<tr>
<td>(option-len octets)</td>
<td></td>
</tr>
</tbody>
</table>
+-----------------------------------------------+
```

Option-code: An unsigned integer identifying the specific option type carried in this option.

Option-len: An unsigned integer giving the length of the option-data field in this option in octets.

Option-data: The data for the option; the format of this data depends on the definition of the option.

GDNP options are scoped by using encapsulation. If an option contains other options, the outer Option-len includes the total size of the encapsulated options, and the latter apply only to the outer option.

3.8.2. Divert Option

The divert option is used to redirect a GDNP request to another node, which may be more appropriate for the intended negotiation or synchronization. It may redirect to an entity that is known as a specific negotiation or synchronization counterpart (on-link or off-link) or a default gateway. The divert option MUST only be encapsulated in Response messages. If found elsewhere, it SHOULD be silently ignored.
Option-code:  OPTION_DIVERT (1).

Option-len:  The total length of diverted destination sub-option(s) in octets.

Locator Option(s) of Diversion Device(s):  Embedded Locator Option(s) (Section 3.8.8) that point to diverted destination device(s).

3.8.3.  Accept Option

The accept option is used to indicate to the negotiation counterpart that the proposed negotiation content is accepted.

The accept option MUST only be encapsulated in Negotiation-ending messages.  If found elsewhere, it SHOULD be silently ignored.

Option-code:  OPTION_ACCEPT (2)

Option-len:  0

3.8.4.  Decline Option

The decline option is used to indicate to the negotiation counterpart the proposed negotiation content is declined and end the negotiation process.

The decline option MUST only be encapsulated in Negotiation-ending messages.  If found elsewhere, it SHOULD be silently ignored.
Option-code:  OPTION_DECLINE (3)
Option-len:  0

Notes: there are scenarios where a negotiation counterpart wants to decline the proposed negotiation content and continue the negotiation process. For these scenarios, the negotiation counterpart SHOULD use a Negotiate message, with either an objective option that contains at least one data field with all bits set to 1 to indicate a meaningless initial value, or a specific objective option that provides further conditions for convergence.

3.8.5. Waiting Time Option

The waiting time option is used to indicate that the negotiation counterpart needs to wait for a further negotiation response, since the processing might need more time than usual or it might depend on another triggered negotiation.

The waiting time option MUST only be encapsulated in Confirm-waiting messages. If found elsewhere, it SHOULD be silently ignored. When received, its value overwrites the negotiation timer (Section 3.7.4).

The counterpart SHOULD send a Negotiation, Negotiation-Ending or another Confirm-waiting message before the negotiation timer expires. If not, the initiator MUST abandon or restart the negotiation procedure, to avoid an indefinite wait.

Option-code:  OPTION_WAITING (4)
Option-len:  4, in octets
Time:  Time in milliseconds
3.8.6. Certificate Option

The Certificate option carries the certificate of the sender. The format of the Certificate option is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION Certificate      |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
```

Option-code: OPTION_CERT_PARAMETER (5)

Option-len: Length of certificate in octets

Public key: A variable-length field containing a certificate

3.8.7. Signature Option

The Signature option allows public key-based signatures to be attached to a GDNP message. The Signature option is REQUIRED in every GDNP message and could be any place within the GDNP message. It protects the entire GDNP header and options. A TimeStamp has been integrated in the Signature Option for anti-replay protection. The format of the Signature option is described as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     OPTION_SIGNATURE          |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           HA-id               |            SA-id              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Timestamp (64-bit)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
|                                                               |
```

Option-code: OPTION_SIGNATURE (6)
Option-len:  12 + Length of Signature field in octets.

HA-id:  Hash Algorithm id.  The hash algorithm is used for computing the signature result. This design is adopted in order to provide hash algorithm agility. The value is from the Hash Algorithm for GDNP registry in IANA. The initial value assigned for SHA-1 is 0x0001.

SA-id:  Signature Algorithm id. The signature algorithm is used for computing the signature result. This design is adopted in order to provide signature algorithm agility. The value is from the Signature Algorithm for GDNP registry in IANA. The initial value assigned for RSASSA-PKCS1-v1_5 is 0x0001.

Timestamp:  The current time of day (NTP-format timestamp [RFC5905] in UTC (Coordinated Universal Time), a 64-bit unsigned fixed-point number, in seconds relative to 0h on 1 January 1900.). It can reduce the danger of replay attacks.

Signature:  A variable-length field containing a digital signature. The signature value is computed with the hash algorithm and the signature algorithm, as described in HA-id and SA-id. The signature constructed by using the sender’s private key protects the following sequence of octets:

1.  The GDNP message header.

2.  All GDNP options including the Signature option (fill the signature field with zeroes).

The signature field MUST be padded, with all 0, to the next 16 bit boundary if its size is not an even multiple of 8 bits. The padding length depends on the signature algorithm, which is indicated in the SA-id field.

3.8.8.  Locator Options

These locator options are used to present a device’s or interface’s reachability information. They are Locator IPv4 Address Option, Locator IPv6 Address Option and Locator FQDN (Fully Qualified Domain Name) Option.

Note that it is assumed that all locators are in scope throughout the GDNP domain. GDNP is not intended to work across disjoint addressing or naming realms.
3.8.8.1. Locator IPv4 address option

Option-code:  OPTION_LOCATOR_IPV4ADDR (7)
Option-len:  4, in octets
IPv4-Address:  The IPv4 address locator of the device/interface
Note: If an operator has internal network address translation for IPv4, this option MUST NOT be used within the Divert option.

3.8.8.2. Locator IPv6 address option

Option-code:  OPTION_LOCATOR_IPV6ADDR (8)
Option-len:  16, in octets
IPv6-Address:  The IPv6 address locator of the device/interface
Note: A link-local IPv6 address MUST NOT be used when this option is used within the Divert option.

3.8.8.3. Locator FQDN option
Option-code: OPTION_FQDN (9)

Option-len: Length of Fully Qualified Domain Name in octets

Domain-Name: The Fully Qualified Domain Name of the entity

Note: Any FQDN which might not be valid throughout the network in question, such as a Multicast DNS name [RFC6762], MUST NOT be used when this option is used within the Divert option.

3.9. Objective Options

3.9.1. Format of Objective Options

An objective option is used to identify objectives for the purposes of discovery, negotiation or synchronization. All objectives must follow a common format as follows:

Option-code: OPTION_XXX: The option code assigned in the specification of the XXX objective.

option-len: The total length in octets.

loop-count: The loop count. This field is present if and only if the objective is a negotiation objective.

flags: Flag bits. This field is present if and only if defined in the specification of the objective.
value: This field is to express the actual value of a negotiation or synchronization objective. Its format is defined in the specification of the objective and may be a single value or a data structure of any kind.

3.9.2. General Considerations for Objective Options

Objective Options MUST be assigned an option type greater than 64 in the GDNP option table.

An Objective Option that contains no additional fields, i.e., has a length of 4 octets, is a discovery objective and MUST only be used in Discovery and Response messages.

The Negotiation Objective Options contain negotiation objectives, which are various according to different functions/services. They MUST be carried by Discovery, Request or Negotiation Messages only. The negotiation initiator MUST set the initial "loop-count" to a value specified in the specification of the objective or, if no such value is specified, to GDNP_DEF_LOOPCT.

For most scenarios, there should be initial values in the negotiation requests. Consequently, the Negotiation Objective options MUST always be completely presented in a Request message, or in a Discovery message in rapid mode. If there is no initial value, the bits in the value field SHOULD all be set to 1 to indicate a meaningless value, unless this is inappropriate for the specific negotiation objective.

Synchronization Objective Options are similar, but MUST be carried by Discovery, Request or Response messages only. They include value fields only in Response messages.

3.9.3. Organizing of Objective Options

As noted earlier, one negotiation objective is handled by each GDNP negotiation thread. Therefore, a negotiation objective, which is based on a specific function or action, SHOULD be organized as a single GDNP option. It is NOT RECOMMENDED to organize multiple negotiation objectives into a single option, nor to split a single function or action into multiple negotiation objectives.

A synchronization objective SHOULD also be organized as a single GDNP option.

Some objectives will support more than one operational mode. An example is a negotiation objective with both a "dry run" mode (where the negotiation is to find out whether the other end can in fact make
the requested change without problems) and a "live" mode. Such modes will be defined in the specification of such an objective. These objectives SHOULD include a "flags" octet, with bits indicating the applicable mode(s).

An objective may have multiple parameters. Parameters can be categorized into two classes: the obligatory ones presented as fixed fields; and the optional ones presented in TLV sub-options or some other form of data structure. The format might be inherited from an existing management or configuration protocol, the objective option acting as a carrier for that format. The data structure might be defined in a formal language, but that is a matter for the specifications of individual objectives. There are many candidates, according to the context, such as ABNF, RBNF, XML Schema, possibly YANG, etc. The GDNP protocol itself is agnostic on these questions.

It is NOT RECOMMENDED to split parameters in a single objective into multiple options, unless they have different response periods. An exception scenario may also be described by split objectives.

3.9.4. Vendor Specific Objective Options

Option codes 128~159 have been reserved for vendor specific options. Multiple option codes have been assigned because a single vendor might use multiple options simultaneously. These vendor specific options are highly likely to have different meanings when used by different vendors. Therefore, they SHOULD NOT be used without an explicit human decision and SHOULD NOT be used in unmanaged networks such as home networks.

There is one general requirement that applies to all vendor specific options. They MUST start with a field that uniquely identifies the enterprise that defines the option, in the form of a registered 32 bit Private Enterprise Number (PEN) [I-D.liang-iana-pen]. There is no default value for this field. Note that it is not used during discovery. It MUST be verified during negotiation or synchronization.

In the case of a vendor-specific objective, the loop count and flags, if present, follow the PEN.
Option-code: OPTION_vendor (128˜159)

Option-len: The total length in octets.

PEN: Private Enterprise Number.

loop-count: The loop count. This field is present if and only if the objective is a negotiation objective.

flags: Flag bits. This field is present if and only if defined in the specification of the objective.

value: This field is to express the actual value of a negotiation or synchronization objective. Its format is defined in the vendor’s specification of the objective.

3.9.5. Experimental Objective Options

Option code 176˜191 have been reserved for experimental options. Multiple option codes have been assigned because a single experiment may use multiple options simultaneously. These experimental options are highly likely to have different meanings when used for different experiments. Therefore, they SHOULD NOT be used without an explicit human decision and SHOULD NOT be used in unmanaged networks such as home networks.

These option codes are also RECOMMENDED for use in documentation examples.

4. Items for Future Work

There are various design questions that are worthy of more work in the near future, as listed below (statically numbered for reference purposes):
1. UDP vs TCP: For now, this specification suggests UDP and TCP as message transport mechanisms. This is not clarified yet. UDP is good for short conversations, is necessary for multicast discovery, and generally fits the discovery and divert scenarios well. However, it will cause problems with large messages. TCP is good for stable and long sessions, with a little bit of time consumption during the session establishment stage. If messages exceed a reasonable MTU, a TCP mode will be required in any case. This question may be affected by the security discussion.

2. DTLS or TLS vs built-in security mechanism. For now, this specification has chosen a PKI based built-in security mechanism based on asymmetric cryptography. However, (D)TLS might be chosen as security solution to avoid duplication of effort. It also allows essentially similar security for short messages over UDP and longer ones over TCP. The implementation trade-offs are different. The current approach requires expensive asymmetric cryptographic calculations for every message. (D)TLS has startup overheads but cheaper crypto per message. DTLS is less mature than TLS.

The following open issues apply only if the current security model is retained:

* 2.1. For replay protection, GDNP currently requires every participant to have an NTP-synchronized clock. Is this OK for low-end devices, and how does it work during device bootstrapping? We could take the Timestamp out of signature option, to become an independent and OPTIONAL (or RECOMMENDED) option.

* 2.2. The Signature Option (Section 3.8.7) states that this option could be any place in a message. Wouldn’t it be better to specify a position (such as the end)? That would be much simpler to implement.

3. DoS Attack Protection needs work.

4. Should we consider a distributed or centralised DNS-like approach to discovery (after the initial discovery needed for bootstrapping)? This topic is deferred for now, but the following considerations apply: This could be a complementary mechanism for multicast based discovery, especially for a very large autonomic network. Centralized registration could be automatically deployed incrementally. At the very first stage, the repository could be empty; then it could be filled in by the objectives discovered by different devices (for example using Dynamic DNS Update). The more records are stored in the repository, the less the multicast-
based discovery is needed. However, if we adopt such a mechanism, there would be challenges: stateful solution, and security.

5. Need to expand description of the minimum requirements for the specification of an individual discovery, synchronization or negotiation objective.

6. Use case and protocol walkthrough. A description of how a node starts up, performs discovery, and conducts negotiation and synchronization for a sample use case would help readers to understand the applicability of this specification. Maybe it should be an artificial use case or maybe a simple real one. However, the authors have not yet decided whether to have a separate document or have it in this document.

7. Cross-check against other ANIMA WG documents for consistency and gaps.

5. Security Considerations

It is obvious that a successful attack on negotiation-enabled nodes would be extremely harmful, as such nodes might end up with a completely undesirable configuration that would also adversely affect their peers. GDNP nodes and messages therefore require full protection.

- Authentication

A cryptographically authenticated identity for each device is needed in an autonomic network. It is not safe to assume that a large network is physically secured against interference or that all personnel are trustworthy. Each autonomic device should be capable of proving its identity and authenticating its messages. GDNP adopts a certificate-based security mechanism to support authentication and data integrity protection.

The timestamp mechanism provides an anti-replay function.

Since GDNP is intended to be deployed in a single administrative domain operating its own trust anchor and CA, there is no need for a trusted public third party.

- Privacy and confidentiality

Generally speaking, no personal information is expected to be involved in the negotiation protocol, so there should be no direct impact on personal privacy. Nevertheless, traffic flow paths, VPNs, etc. could be negotiated, which could be of interest for
traffic analysis. Also, operators generally want to conceal details of their network topology and traffic density from outsiders. Therefore, since insider attacks cannot be excluded in a large network, the security mechanism for the protocol MUST provide message confidentiality.

- DoS Attack Protection

TBD.

- Security during bootstrap and discovery

A node cannot authenticate GDNP traffic from other nodes until it has identified the trust anchor and can validate certificates for other nodes. Also, until it has successfully enrolled [I-D.pritikin-anima-bootstrapping-keyinfra] it cannot assume that other nodes are able to authenticate its own traffic. Therefore, GDNP discovery during the bootstrap phase for a new device will inevitably be insecure and GDNP synchronization and negotiation will be impossible until enrollment is complete.

6. IANA Considerations

Section 3.4 defines the following multicast addresses, which have been assigned by IANA for use by GDNP:

ALL_GDNP_NEIGHBOR multicast address (IPv6): (TBD1)

ALL_GDNP_NEIGHBOR multicast address (IPv4): (TBD2)

Section 3.4 defines the following UDP and TCP port, which has been assigned by IANA for use by GDNP:

GDNP Listen Port: (TBD3)

This document defined a new General Discovery and Negotiation Protocol. The IANA is requested to create a new GDNP registry. The IANA is also requested to add two new registry tables to the newly-created GDNP registry. The two tables are the GDNP Messages table and GDNP Options table.

Initial values for these registries are given below. Future assignments are to be made through Standards Action or Specification Required [RFC5226]. Assignments for each registry consist of a type code value, a name and a document where the usage is defined.
GDNP Messages table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in Section 3.7 in this document:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>RFCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>this document</td>
</tr>
<tr>
<td>1</td>
<td>Discovery</td>
<td>this document</td>
</tr>
<tr>
<td>2</td>
<td>Response</td>
<td>this document</td>
</tr>
<tr>
<td>3</td>
<td>Request Message</td>
<td>this document</td>
</tr>
<tr>
<td>4</td>
<td>Negotiation Message</td>
<td>this document</td>
</tr>
<tr>
<td>5</td>
<td>Negotiation-end Message</td>
<td>this document</td>
</tr>
<tr>
<td>6</td>
<td>Confirm-waiting Message</td>
<td>this document</td>
</tr>
</tbody>
</table>

GDNP Options table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in Section 3.8 and Section 3.9.1 in this document:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>RFCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td>this document</td>
</tr>
<tr>
<td>1</td>
<td>Divert Option</td>
<td>this document</td>
</tr>
<tr>
<td>2</td>
<td>Accept Option</td>
<td>this document</td>
</tr>
<tr>
<td>3</td>
<td>Decline Option</td>
<td>this document</td>
</tr>
<tr>
<td>4</td>
<td>Waiting Time Option</td>
<td>this document</td>
</tr>
<tr>
<td>5</td>
<td>Certificate Option</td>
<td>this document</td>
</tr>
<tr>
<td>6</td>
<td>Signature Option</td>
<td>this document</td>
</tr>
<tr>
<td>7</td>
<td>Device IPv4 Address Option</td>
<td>this document</td>
</tr>
<tr>
<td>8</td>
<td>Device IPv6 Address Option</td>
<td>this document</td>
</tr>
<tr>
<td>9</td>
<td>Device FQDN Option</td>
<td>this document</td>
</tr>
<tr>
<td>10~63</td>
<td>Reserved for future GDNP</td>
<td></td>
</tr>
<tr>
<td>64~127</td>
<td>Reserved for future GDNP</td>
<td></td>
</tr>
<tr>
<td>128~159</td>
<td>Vendor Specific Options</td>
<td>this document</td>
</tr>
<tr>
<td>160~175</td>
<td>Reserved for future use</td>
<td></td>
</tr>
<tr>
<td>176~191</td>
<td>Experimental Options</td>
<td>this document</td>
</tr>
<tr>
<td>192~65535</td>
<td>Reserved for future use</td>
<td></td>
</tr>
</tbody>
</table>

The IANA is also requested to create two new registry tables in the GDNP Parameters registry. The two tables are the Hash Algorithm for GDNP table and the Signature Algorithm for GDNP table.

Initial values for these registries are given below. Future assignments are to be made through Standards Action or Specification Required [RFC5226]. Assignments for each registry consist of a name, a value and a document where the algorithm is defined.
Hash Algorithm for GDNP. The values in this table are 16-bit unsigned integers. The following initial values are assigned for Hash Algorithm for GDNP in this document:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>RFCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0x0000</td>
<td>this document</td>
</tr>
<tr>
<td>SHA-1</td>
<td>0x0001</td>
<td>this document</td>
</tr>
<tr>
<td>SHA-256</td>
<td>0x0002</td>
<td>this document</td>
</tr>
</tbody>
</table>

Signature Algorithm for GDNP. The values in this table are 16-bit unsigned integers. The following initial values are assigned for Signature Algorithm for GDNP in this document:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>RFCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0x0000</td>
<td>this document</td>
</tr>
<tr>
<td>RSASSA-PKCS1-v1_5</td>
<td>0x0001</td>
<td>this document</td>
</tr>
</tbody>
</table>

7. Acknowledgements

A major contribution to the original version of this document was made by Sheng Jiang.

Valuable comments were received from Michael Behringer, Zongpeng Du, Yu Fu, Zhenbin Li, Dimitri Papadimitriou, Michael Richardson, Markus Stenberg, Rene Struik, Dacheng Zhang, and other participants in the NMRG research group and the ANIMA working group.

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

8. Change log [RFC Editor: Please remove]

draft-carpenter-anima-discovery-negotiation-protocol-02, 2015-02-19:

Tuned requirements to clarify scope,
Clarified relationship between types of objective,
Clarified that objectives may be simple values or complex data structures,
Improved description of objective options,
Added loop-avoidance mechanisms (loop count and default timeout, limitations on discovery relaying and on unsolicited responses),
Allow multiple discovery objectives in one response,
Provided for missing or multiple discovery responses,
Indicated how modes such as "dry run" should be supported,
Minor editorial and technical corrections and clarifications,
Reorganized future work list.

draft-carpenter-anima-discovery-negotiation-protocol-01, restructured
the logical flow of the document, updated to describe synchronization
completely, add unsolicited responses, numerous corrections and
clarifications, expanded future work list, 2015-01-06.

draft-carpenter-anima-discovery-negotiation-protocol-00, combination
of draft-jiang-config-negotiation-ps-03 and draft-jiang-config-
negotiation-protocol-02, 2014-10-08.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1
(SHA1)", RFC 3174, September 2001.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness

[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

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko,

9.2. Informative References

[I-D.behringer-anima-autonomic-control-plane]
Behringer, M., Bjarnason, S., Bl, B., and T. Eckert, "An
Autonomic Control Plane", draft-behringer-anima-autonomic-
control-plane-00 (work in progress), October 2014.
[I-D.chaparadza-intarea-igcp]

[I-D.eckert-anima-stable-connectivity]

[I-D.ietf-dnssd-requirements]
Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-SD/mDNS Extensions", draft-ietf-dnssd-requirements-04 (work in progress), October 2014.

[I-D.ietf-homenet-dncp]

[I-D.ietf-homenet-hncp]

[I-D.ietf-netconf-restconf]

[I-D.irtf-nmrg-an-gap-analysis]

[I-D.irtf-nmrg-autonomic-network-definitions]
[I-D.liang-iana-pen]
Liang, P., Melnikov, A., and D. Conrad, "Private
Enterprise Number (PEN) practices and Internet Assigned
Numbers Authority (IANA) registration considerations",
draft-liang-iana-pen-04 (work in progress), July 2014.

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

Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1

"Service Location Protocol, Version 2", RFC 2608, June
1999.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson,
"Remote Authentication Dial In User Service (RADIUS)", RFC
2865, June 2000.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V.,
and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP

and M. Carney, "Dynamic Host Configuration Protocol for

[RFC3416] Presuhn, R., "Version 2 of the Protocol Operations for the
Simple Network Management Protocol (SNMP)", STD 62, RFC
3416, December 2002.


[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman,
"Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
September 2007.
Appendix A. Capability Analysis of Current Protocols

This appendix discusses various existing protocols with properties related to the above negotiation and synchronisation requirements. The purpose is to evaluate whether any existing protocol, or a simple combination of existing protocols, can meet those requirements.

Numerous protocols include some form of discovery, but these all appear to be very specific in their applicability. Service Location Protocol (SLP) [RFC2608] provides service discovery for managed networks, but requires configuration of its own servers. DNS-SD [RFC6763] combined with mDNS [RFC6762] provides service discovery for small networks with a single link layer. [I-D.ietf-dnssd-requirements] aims to extend this to larger autonomous networks. However, both SLP and DNS-SD appear to target primarily application layer services, not the layer 2 and 3 objectives relevant to basic network configuration.

Routing protocols are mainly one-way information announcements. The receiver makes independent decisions based on the received
information and there is no direct feedback information to the
announcing peer. This remains true even though the protocol is used
in both directions between peer routers; there is state
synchronization, but no negotiation, and each peer runs its route
calculations independently.

Simple Network Management Protocol (SNMP) [RFC3416] uses a command/
response model not well suited for peer negotiation. Network
Configuration Protocol (NETCONF) [RFC6241] uses an RPC model that
does allow positive or negative responses from the target system, but
this is still not adequate for negotiation.

There are various existing protocols that have elementary negotiation
abilities, such as Dynamic Host Configuration Protocol for IPv6
(DHCPv6) [RFC3315], Neighbor Discovery (ND) [RFC4861], Port Control
Protocol (PCP) [RFC6887], Remote Authentication Dial In User Service
(RADIUS) [RFC2865], Diameter [RFC6733], etc. Most of them are
configuration or management protocols. However, they either provide
only a simple request/response model in a master/slave context or
very limited negotiation abilities.

There are also signalling protocols with an element of negotiation.
For example Resource ReSerVation Protocol (RSVP) [RFC2205] was
designed for negotiating quality of service parameters along the path
of a unicast or multicast flow. RSVP is a very specialised protocol
aimed at end-to-end flows. However, it has some flexibility, having
been extended for MPLS label distribution [RFC3209]. A more generic
design is General Internet Signalling Transport (GIST) [RFC5971], but
it is complex, tries to solve many problems, and is also aimed at
per-flow signalling across many hops rather than at device-to-device
signalling. However, we cannot completely exclude extended RSVP or
GIST as a synchronization and negotiation protocol. They do not
appear to be directly useable for peer discovery.

We now consider two protocols that are works in progress at the time
of this writing. Firstly, RESTCONF [I-D.ietf-netconf-restconf] is a
protocol intended to convey NETCONF information expressed in the YANG
language via HTTP, including the ability to transit HTML
intermediaries. While this is a powerful approach in the context of
centralised configuration of a complex network, it is not well
adapted to efficient interactive negotiation between peer devices,
especially simple ones that are unlikely to include YANG processing
already.

Secondly, we consider Distributed Node Consensus Protocol (DNCP)
[I-D.ietf-homenet-dncp]. This is defined as a generic form of state
synchronization protocol, with a proposed usage profile being the

Specific features of DNCP include:

- Every participating node has a unique node identifier.
- DNCP messages are encoded as a sequence of TLV objects, sent over unicast UDP or TCP, with or without (D)TLS security.
- Multicast, if available, is used only for discovery of DNCP neighbors when lower security is acceptable.
- Synchronization of state is maintained by the Trickle algorithm. There is no negotiation capability.
- The HNCP profile of DNCP is designed to operate between directly connected neighbors on a shared link using UDP and link-local IPv6 addresses.

Clearly DNCP does not meet the needs of a general negotiation protocol, especially in its HNCP profile due to the limitation to link-local messages and its strict dependency on IPv6. However, at the minimum it is a very interesting test case for this style of interaction between devices without needing a central authority.

A proposal was made some years ago for an IP based Generic Control Protocol (IGCP) [I-D.chaparadza-intarea-igcp]. This was aimed at information exchange and negotiation but not directly at peer discovery. However, it has many points in common with the present work.

None of the above solutions appears to completely meet the needs of generic discovery, state synchronization and negotiation in a single solution. Neither is there an obvious combination of protocols that does so. Therefore, this document proposes the design of a protocol that does meet those needs. However, this proposal needs to be compared with alternatives such as extension and adaptation of GIST or DNCP, or combination with IGCP.

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

Email: brian.e.carpenter@gmail.com

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

Email: leo.liubing@huawei.com
Abstract

This document presents the importance of security bootstrapping for IoT networks, analyzes the state-of-the-art works in standard organizations and discusses what should be considered when designing the secure bootstrapping mechanism.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

1. Introduction ........................................... 2
2. Terminology ............................................ 3
3. Analysis of Related State-of-the-art Works ............ 3
   3.1. Security bootstrapping ............................. 3
       3.1.1. Authentication framework .................... 4
       3.1.2. Credential Material and Architecture ........ 7
   3.2. Higher Layer Protocol Use After/During Bootstrapping . 9
4. Role of IoT Security Bootstrapping .................... 10
5. Design Considerations .................................. 10
   5.1. Able to clearly define security dependency and trust domains ................................ 12
   5.2. Cross-layer design ................................ 12
   5.3. Reduce human interaction to the minimum ............ 12
   5.4. Able to resist attacks ................................ 13
   5.5. Low computation cost and communication overhead ........ 13
6. Security Considerations .................................. 13
7. Acknowledgements ....................................... 13
8. References ............................................. 13
   8.1. Normative References ............................... 13
   8.2. Informative References .............................. 15
Authors’ Addresses ......................................... 17

1. Introduction

An Internet of Things (IoT) network is composed of connected things that cooperate together to accomplish tasks such as smart buildings, smart environment monitoring system, intelligent transport system, etc. The size of IoT varies from tens to thousands depending on the application, and things in an IoT network might be produced by different vendors and they are normally heterogeneous with various constraints e.g. power supply, communication capability, CPU and memory.

IEEE 802.15.4 specifies the physical layer and media access control for low-rate wireless personal area networks (LR-WPANs). It is widely used in wireless sensor networks nowadays and is foreseen as the most used lower layer protocol for low rate IoT networks with resource constrained devices. In IETF, 6LoWPAN (concluded) developed RFC 4944 [RFC4944] to describe how to transmit IPv6 packets over 802.15.4, and support mesh routing in LR-WPANs. 6lo defines generic IPv6 packet header compression method [RFC7400] for LR-WPANs. 6tisch tries to build adaptation protocols for IEEE 802.15.4e specification. Roll develops routing protocol RPL [RFC6550] for IPv6 based low power and lossy networks. Note that IEEE 802.15.4 can be applied to mobile nodes, routing protocols such as AODV [RFC3561], DSL [RFC4728], OLSR [RFC3626] by MANET group are also widely used. CoAP [RFC7252] from
CoRE defines a UDP based web transfer protocol for machine-to-machine (M2M) applications such as smart energy and building automation.

The above mentioned protocols provide different selections of IoT protocol stacks to fulfill specific tasks based on IEEE 802.15.4. At the start-up phase of a network or after the provisioned communications have failed, bootstrapping is typically required to configure nodes at all layers, including anything from link-layer information (i.e., wireless channels, link-layer encryption keys) to application-layer information (i.e., network names, application encryption keys). It can be realized either manually via user interface or automatically via interaction between nodes.

Traditional bootstrapping approaches tend to impose configuration burdens upon users. For example, users need to follow a series of instruction steps for configuration. Configuring IoT devices becomes more complicated since they don’t always provide user interface to input all necessary information, and the scale of the IoT network can be large, dynamic or error prone. As a result, human intervention is expensive and not efficient in those situations. This motivates the need for self-organization and automatic self-bootstrapping in IoT. Enabling a plug & play framework not only reduces human efforts in configuring IoT but also improve the scalability and flexibility. This draft presents a survey of the state-of-the-art works on bootstrapping/networking in IETF, ZigBee Alliance, IEEE and Thread group, and the design considerations for security bootstrapping are derived.

2. Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

3. Analysis of Related State-of-the-art Works

Bootstrapping is required at all layers, where different conditions and information should be transferred for different protocols. This section provides analysis on the existing bootstrapping works in standard organizations and summarizes the concerns.

3.1. Security bootstrapping

Security bootstrapping includes the authentication of devices to establish trust relationships in a network, as well as transferring security parameters and keying materials. Security bootstrapping is
believed as the fundamental part of bootstrapping, because once secure and authentic channels are established, the bootstrapping of all other information can be conducted as ordinary secured communications. Accordingly, many works focus on security bootstrapping and device authentication. In IETF, [I-D.pritikin-anima-bootstrapping-keyinfra] is proposed in Anima, [I-D.sarikaya-6lo-bootstrapping-solution] is proposed in 6lo, [I-D.struik-6tisch-security-considerations] is in 6tisch, [I-D.kwatsen-netconf-zerocontacts] is in Netconf, and [I-D.he-iot-security-bootstrapping] is proposed for bootstrapping IEEE 802.15.4 based IoT networks. ZigBee IP stack is developed by ZigBee Alliance and it supports EAP-TLS and PANA as authentication protocols. In Thread Group, a networking solution is developed. The devices are authenticated through pre-installed codes. IEEE 802.15.4 also defines two-step mechanism for nodes joining network with layer 2 authentication without considering IP infrastructure.

3.1.1. Authentication framework

The arguments on authentication framework focus on EAP, PANA, HIP-DEX, 802.1X via EAPOL, and IKEv2.

[I-D.oflynn-core-bootstrapping] relates the aforementioned authentication frameworks into IEEE 802.15.4 and requirements in order to use them for bootstrapping procedure.

- If PANA is used, a new entity called PANA Relay Element should be added in the architecture and behavior of PANA RE needs to be defined [RFC6345]; New AVPs needed for PANA Relay Element operation for relaying messages from the client to the authenticator and vice versa are required to be specified. If PANA is used to securely distribute group key [RFC6786] from the PANA Authentication Agent to the PANA Client using AES Key Wrap with padding algorithm, an extension to PANA needs to be defined.

- If HIP-DEX is used, the initiator should be able to get the IP address of the responder, either using DNS infrastructure or local configuration.

- If 802.1X is used, a special value in the Frame Type subfield of the Frame Control Field of IEEE 802.15.4 MAC header should be assigned to indicate the type of the payload. Group addresses for 802.15.4 corresponding to EAPOL Group Address Assignments defined in Table 11.1 of [IEEE802.1x] are required, especially for EAPOL-Start packet. The mapping of MAC frames and security level to different types should be defined, for instance: which MAC frames of beacon, data, acknowledgment and MAC command as defined in [IEEE802.15.4] with what security levels are mapped to controlled
port, which MAC frames with what security levels are mapped to uncontrolled port and which MAC frames are never mapped to any of controlled/uncontrolled port (i.e., the payload of those frames are used by the MAC-layer itself and never used by upper layers).

[I-D.garcia-core-security] discusses about using Internet Key Exchange protocol version 2 (IKEv2) as authentication method. It summarizes that IKEv2 can perform key exchanges and the setup of security associations without online connections to a trust center. It provides end-to-end security, and supports host mobility with MOBIKE extension. However, MOBIKE mandates the use of IPsec tunnel mode which requires to transmit an additional IP header in each packet. This additional overhead could be alleviated by using header compression methods or the Bound End-to-End Tunnel (BEET) mode [I-D.nikander-esp-beet-mode], a hybrid of tunnel and transport mode with smaller packet headers.

Several EAP methods have been standardized for different purposes. One widely used method is the EAP-TLS [RFC7250] which enables mutual authentication and distribute keying material to secure subsequent communications. However it only supports certificate-based mutual authentication, thus public key infrastructure is required and fragmentation is needed when using IEEE 802.15.4 to exchange authentication messages.

ZigBee Alliance specified an IPv6 stack aimed at IEEE 802.15.4 devices mainly used in smart meters developed primarily for SEP 2.0 (Smart Energy Profile) application layer traffic [SEP2.0]. This specification assumes Class 2 devices which have 50 KiB of RAM and 250 KiB of flash memory [RFC7228]. Some devices in such systems have more resources and processing power (e.g. ARM9 core, MiBs RAM/Flash). For security bootstrapping, ZigBee IP uses EAP-TLS. Authentication that is not based on certificates reduces cost of certificate management and fewer messages are needed to be exchanged between client and server. [I-D.sarikaya-6lo-bootstrapping-solution] proposes to use raw public keys via EAP-TLS, thus extension to EAP-TLS is indicated. Note that EAP requires exchanging the device identity in plain text at the beginning, but how to protect the privacy information indicated in the device ID is out of concern of EAP methods.

EAP-PSK [RFC4764] is another EAP method. It realizes mutual authentication and session key derivation using a Pre-Shared Key (PSK). Normally four messages are exchanged in the authentication process. Once the authentication is successful, EAP-PSK provides a protected communication channel.
EAP-IKEv2 [RFC5106] is an EAP method based on IKEv2. It provides mutual authentication and session key establishment between an EAP peer and an EAP server. It supports authentication techniques that are based on different credentials including asymmetric key pairs, symmetric keys and passwords. Besides, it is possible to use a different authentication credential in each direction. For example, the EAP server authenticates itself using public/private key pair and the EAP peer using symmetric key. As a result different combinations of credentials are expected to be used in practice. Compared with EAP-TLS and EAP-PSK, EAP-IKEv2 supports mobility and different authentication techniques.

[I-D.kumar-6lo-selective-bootstrap] presents a selective bootstrapping/commissioning method by introducing the concept of Commissioning Tool (CT). In this method the devices are let to connect to the network and execute 6LowPAN neighbor discovery protocol and have an IPv6 address before they are authenticated. Then the devices are selected one by one in some order to communicate with the CT via untrusted constructed route. Once the ID of joining device is authenticated, CT sends the layer-2 key material to the device via secured channel, which is established by DTLS by exchanging credential material installed during manufacturing.

The bootstrapping method in [I-D.kumar-6lo-selective-bootstrap] creates security risks for the network by

1. letting the devices have IP addresses for layer3 communication before authentication.
2. constructing routing topology before devices are authenticated.
3. establishing transport layer security before layer-2 security.

However, such a protocol could be justified in some application domains like lightning control systems.

There is work going on in the IEEE 802.15.9 task group which specifies a way to transport existing key management protocols (KMP) over the 802.15.4 frames. The new feature would allow running IKEv2, EAP, PANA, 802.1X, HIP and Dragonfly over the IEEE 802.15.4 and generate keys for 802.15.4 security and protect all messages between the two nodes [IEEE802.15.9]. It would be desired if the security bootstrapping procedure reuses the KMPs that supported by lower layers to reduce cost.

Table 1 summarizes the authentication frameworks and credential materials of the aforementioned solutions.
<table>
<thead>
<tr>
<th>Referenced solution</th>
<th>Authentication method</th>
<th>Credential material</th>
</tr>
</thead>
<tbody>
<tr>
<td>[I-D.pritikin-anima-bootstrapping-keyinfra]</td>
<td>802.1x-EAPOL, EAP-TLS, EAP-IKEv2</td>
<td>802.1AR certificate</td>
</tr>
<tr>
<td>[I-D.sarikaya-6lo-bootstrapping-solution]</td>
<td>EAP-TLS (modified)</td>
<td>Raw public key</td>
</tr>
<tr>
<td>[I-D.kwatsen-netconf-zerotouch]</td>
<td>Unspecified (EAP-TLS might be used)</td>
<td>X.509 certificate</td>
</tr>
<tr>
<td>[I-D.he-iot-security-bootstrapping]</td>
<td>EAP, PANA</td>
<td>Unspecified</td>
</tr>
<tr>
<td>[I-D.kumar-6lo-selective-bootstrap]</td>
<td>Selected by Commissioner with CT</td>
<td>PSK defined in CT Certificate</td>
</tr>
<tr>
<td>ZigBee IP stack based Smart Energy</td>
<td>EAP-TLS, PANA</td>
<td>Certificate</td>
</tr>
<tr>
<td>Thread networking</td>
<td>Unknown</td>
<td>Product install codes</td>
</tr>
</tbody>
</table>

### 3.1.2. Credential Material and Architecture

The trust relationship can be established by exchanging credential materials, which can be asymmetric with user authentication or with certificate authority, or symmetric pre-shared key configured by network developer. In certificate authority (CA), a typical public key infrastructure (PKI) is used, meaning that a set of hardware, software, people, policies, and procedures are needed to create, manage, distribute, use, store, and revoke digital certificates. The public keys are obtained in PKI containers, and both ends are validated using trust anchors based on a certification authority (CA). [I-D.pritikin-anima-bootstrapping-keyinfra] uses 802.1AR certificate, [I-D.kwatsen-netconf-zerotouch] uses X.509 certificate. Certificate mechanism provides high security however it can add a complicated trust relationship that is difficult to validate. When it comes to large scale IoT networks, certificate management and distribution will raise scalability and flexibility issue. Besides, the time spent and CPU occupied by the cryptographic operations is non-trivial when this mechanism is implemented on computational devices.
constraint devices. Since some IEEE802.15.4 technologies including
802.15.4e only allows 127 Octets maximum payload, fragmentation is
unavoidable, which indicates that a large amount of data is
transmitted and communication overhead is heavy. The public-key
based handshake process of EAP-TLS is part of the bottleneck that
significantly degrades the performance. Designers are forced to use
highly efficient protocols for the sake of ensuring the computational
complexity of security algorithms as low as possible.

In today’s IoT, most common architectures are fully centralized in a
sense that all the security relationships within a segment are
handled by a central party.

The 802.1x framework, the architecture proposed in
[I-D.pritikin-anima-bootstrapping-keyinfra] and the ZigBee IP smart
energy solution are centralized. A centralized authentication
architecture allows for central management of devices and keying
materials as well as for the backup of cryptographic keys. As a
result there is no high requirement on network devices in a
centralized architecture. However it also represents a single point
of failure and is more suitable for static network where the route to
the trust center/AAA server is stable.

The self-signed certificates are commonly used in smaller deployments
where they are distributed to all involved protocol endpoints out-of-
band, thus CA and certificate management are not required. This
practice does, however, still require the overhead of the certificate
generation even though none of the information found in the
certificate is actually used.

The raw public key method is proposed to generate light weight
certificate, which can significantly reduce overhead. However, the
self-signed certificate and raw public key only prove the possession
of the private-public key pair and are unable to prove whether the
owner is legitimate.

The pre-shared key based mechanisms are more suitable for constrained
environments, e.g. wireless communications, and limited CPU power
devices. It enables mutual authentication, meanwhile requires less
cryptographic operations and less communication overhead compared
with certificate based mechanism. However, traditional approaches of
key generation/distribution tend to impose configuration burdens upon
users. For example, users need to follow a series of instruction
steps for WiFi Protected Access 2, Pre-shared key (WPA2-PSK)
configuration, even though the pre-shared key mode is the simplest
option for using WPA. Establishing security among IoT devices
becomes more complicated since they don’t always provide user
interface to input necessary security information.
As discussed, the authentication of self-signed certificate and pre-shared key mechanisms are distributed. Distributed architecture allows creating ad-hoc security domains that might not require a single online management entity and are operative in a much more stand-alone manner. In this case, hardware should be configured to be able to authenticate and verify other peers.

In today's IoT, most common architectures are fully centralized in a sense that all the security relationships within a segment are handled by a central party.

The Thread protocol is expected to use product install codes as authentication material. Currently not enough details are available on the Thread protocol.

Physical unclonable function (PUF) arises as a promising authentication technology. PUF is a physical entity that is embodied in a physical structure and is easy to evaluate but hard to predict. Further, an individual PUF device must be easy to make but practically impossible to duplicate, even given the exact manufacturing process that produced it. In this respect it is the hardware analog of a one-way function. PUFs can serve as a root of the trust and can provide a key which cannot be easily reverse engineered. Temperature and aging have been given special attention on developing reliable PUF [MIT2014].

3.2. Higher Layer Protocol Use After/During Bootstrapping

Configurations of parameters for other protocols are important as well to ensure a successful networking. Those parameters are transferred upon a successful security bootstrapping.

The IP address configuration is a major issue which must be solved before any other higher layer service can start. It can be locally pre-configured, auto-configured or managed from a third party tool.

- Pre-configured: is mainly what is done today. No further network service is needed, the assignment is done from a planning/commissioning tool instead. This method requires human interaction, devices with IP configured are trusted by default. scalability and flexibility cannot be satisfied in this case.

- Auto-configuration: the device creates its IP address itself, applying one of the algorithms specified in the relevant standards, e.g. ZigBee IP solved this problem by using SLAAC IPv6 addresses based on the EUI-64; [I-D.pritikin-anima-bootstrapping-keyinfra] suggests to obtain an IP address using existing methods, such as SLAAC or DHCPv6. RPL
[RFC6550] is a special routing protocol that generates for each device an IP prefix based on the constructed routing topology, thus special attention should be paid as chicken/egg issue arises when relay of authentication is needed by the network level bootstrapping. The auto-configured IP address may need to perform a check for duplicates (i.e. APIPA17). Encoding of semantics into the address may need information from lower layer (see above) or from network service. Note, this only works for so-called link local-addresses which are valid only in one Ethernet domain.

- Managed: pre-planned addresses are assigned by means of a third party database, such as DHCP, a central server.

4. Role of IoT Security Bootstrapping

Figure 1 shows a network life cycle: after IoT devices being deployed in field, the security bootstrapping starts. Devices are authenticated, keying materials are exchanged for securing subsequent configuration/data exchange messages. The device gets an IP address and joins the network.

```
+--------------------------------------+
|          Device deployment           |
+----------------+---------------------+   |
|             ------------+   |
|    Network access authentication     |   |
+----------------+---------------------+   |
|                         +->Security Bootstrapping |
+----------------+---------------------+   |
|    Secured channel keying material   |   |
+----------------+---------------------+   |
|             ------------+   |
|    Secure communication in the network |
+------------------------------------------+
```

Figure 1: IoT Life Cycle

5. Design Considerations

IoT can be deployed in different environments for different applications, which calls for protocols with options where a set of options is selected to construct a protocol stack that fits for a given environment, e.g. home, enterprise or industrial. The deployment and configurations can also be divided into two types, one is for static network, and the other is for dynamic network.
IoT developed in buildings, homes, or industrial areas are often static. A general approach is that a network engineer plans the locations for each device and determines topology of network based on deployment environment and channel estimation. Then the key devices (e.g., sink nodes, or parent nodes of a routing protocol) are installed before deploying other devices. Upon successful installation, the device is plugged and security bootstrapping is run in either centralized or distributed manner with pre-configured credential material. The device is at work after all the protocols are successfully bootstrapped. When a new device joins an existing network, the joining device bootstrapping procedure is triggered by itself.

In a dynamic network where devices come and go, their IPv6 addresses might also change. Bootstrapping/re-commissioning at network level is more frequently required than that in static network, hence minimum human interaction is highly preferred. Reducing communication overhead will improve the efficiency of networking, and this is especially useful for low bandwidth and low rate IEEE 802.15.4.

Mains-powered devices can stay continuously connected to the network. Normally-off power strategy can be used for battery powered devices where the devices sleep long periods of time and stay disconnected and reattach to the network after it is woken up. Between these two extremes, there is low-power device mode where the devices need to be able to communicate on a relatively frequent basis [RFC7228]. Bootstrapping protocol needs to be able to take into consideration these power levels in the design.

The order of bootstrapping is another concern in designing the bootstrapping protocol. The devices could arbitrarily be bootstrapped as they join the network, especially in dynamic topologies. In static topologies the order could be completely installation and installer dependent and could be optimized to lower cost and could be independent of network topology [I-D.kumar-6lo-selective-bootstrap]. The order is also dependent on the architecture of authentication. For centralized architecture, incremental approach is recommended by [I-D.he-iot-security-bootstrapping], [I-D.garcia-core-security] and [I-D.sarikaya-6lo-bootstrapping-solution], whereas a selective order can be specified by CT [I-D.kumar-6lo-selective-bootstrap] and special attention should be paid on the secured channel establishment via untrusted route. For decentralized architecture, the mutual-authentication is realized between equal peers in pure mesh topology without any preferred order and network keys can be distributed by cluster heads once clusters are formed.
Some mandatory considerations can be derived from different applications for IoT security bootstrapping mechanism:

5.1. Able to clearly define security dependency and trust domains

Things of IoT are more related to private data, thus trust increases its importance. It is easy to introduce a new node in a deployed IoT to capture and analyze the data traffic. As a result,

a. Security dependencies between different devices must be clarified. Circular dependencies must be avoided.

b. The designed protocol should enable mutual authentication between devices running the security bootstrapping protocol. Proper authentication material and mechanism should be chosen.

c. The security bootstrapping protocol processing devices should agree upon the security associations (e.g. key materials, algorithms etc.) for securing their communications before exchanging any protocol packets.

5.2. Cross-layer design

The security bootstrapping method should take into account the features and requirements of full stack protocols that are selected for an IoT network. Security bootstrapping in collaboration with other networking protocols is likely to produce a comprehensive solution.

Cooperative communication and scheduling among neighboring things at lower layer will reduce the possibility of network congestion and assist finishing bootstrapping efficiently. Different power modes should be considered by the designed protocol.

As discussed in Section 3.2, higher layer protocols impact the procedure of bootstrapping. During network start-up, link local IP address should be assigned in order to run PANA/TLS to forward authentication messages by IoT routing protocols such as AODV and DSR in MANET. However, the RPL for LLN configures IP addresses for all the devices during/ at the end of routing procedure, which may create a chicken/egg issue when PANA/TLS are also used. 802.1X uses link layer address so no IP address is needed.

5.3. Reduce human interaction to the minimum

Configuring IoT devices can be complicated since they don’t always provide user interface to input all necessary information, and the scale of the IoT network can be large, dynamic or error prone.
Besides, IoT network users usually do not have expertise in networking, this motivates self-organizing IoT network protocol that start from security bootstrapping. As a result, the design of bootstrapping protocol should be able to reduce human interaction to the minimum.

5.4. Able to resist attacks

The designed bootstrapping protocol should be able to resist attacks and protect CIA triad. Typical threat modeling approaches (e.g. STRIDE) should be used to guide the design of bootstrapping architecture and procedure. STRIDE categorizes attack into spoofing, Tampering with data, Repudiation, Information disclosure, Denial of service and Elevation of privilege.

5.5. Low computation cost and communication overhead

The amount of transmitted data and the complexity of data processing should be optimized to the minimum to save computation and communication cost.

6. Security Considerations

TBD

7. Acknowledgements

TBD

8. References

8.1. Normative References

[IEEE802.15.4]

[IEEE802.15.9]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.


Informative References

[I-D.garcia-core-security]
[I-D.he-iot-security-bootstrapping]
ana.hedanping@huawei.com, a., "Security Bootstrapping of IEEE 802.15.4 based Internet of Things", draft-he-iot-security-bootstrapping-00 (work in progress), January 2015.

[I-D.kumar-6lo-selective-bootstrap]
Kumar, S. and P. Stok, "Security Bootstrapping over IEEE 802.15.4 in selective order", draft-kumar-6lo-selective-bootstrap-00 (work in progress), March 2015.

[I-D.kwatsen-netconf-zerotouch]

[I-D.nikander-esp-beet-mode]

[I-D.oflynn-core-bootstrapping]

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

[I-D.sarikaya-6lo-bootstrapping-solution]

[I-D.struik-6tisch-security-considerations]

Authors’ Addresses

Ana(Danping) He (editor)
Huawei
Building Q14, 156 Beiqing Road
Beijing 100095
China

Email: ana.hedanping@huawei.com

Behcet Sarikaya (editor)
Huawei USA
5340 Legacy Dr. Building 3
Plano, TX 75024

Email: sarikaya@ieee.org
Zero Touch Provisioning for NETCONF Call Home (ZeroTouch)
draft-ietf-netconf-zerotouch-02

Abstract

This draft presents a technique for establishing a secure NETCONF connection between a newly deployed IP-based device, configured with just its factory default settings, and its rightful owner's network management system (NMS).

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

A fundamental business requirement is to reduce costs where possible. For network operators, deploying devices to many locations can be a significant cost, as sending trained specialists to each site to do installations is both cost prohibitive and does not scale.
The solution presented herein enables a device to securely obtain a bootstrapping configuration from the network without any operator input. Significantly, this configuration may configure the device to securely call home using NETCONF Call Home [draft-ietf-netconf-call-home].

Central to this solution is the device being able to process a set of files locally, without any need to reach out to the network again. As consequence, how the files are obtained is not critical to the security of the solution. The files can be read over any networking layer or medium. By example, the files could be loaded using a USB flash drive physically plugged into a device.

The solution presented below focuses on supporting IP networks that may have a DHCP server. Solutions for other deployment scenarios may be defined by drafts in the future.

1.1. Use Cases

- Connecting to a remotely administered network
  
  This use-case involves scenarios, such as a remote branch office or convenience store, whereby the device connects an access gateway device to an ISP’s network. In this case, the device receives only generic networking settings provided by the ISP’s DHCP server, with no site-specific customizations possible. In such a case, the device has no recourse but to reach out to the public Internet its initial configuration.

- Connecting to a locally administered network
  
  This use-case covers all other scenarios and differs only in that the device may additionally receive site-specific information from the network, which could direct it to use a local server for its initial configuration. If no site-specific information is provided, or the device is unable to use the information provided, it can then reach out to network just as it would for a remotely administered network.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in the sections below are to be interpreted as described in RFC 2119 [RFC2119].
1.3. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Braces "+" and "+" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. High-level Design

2.1. Design Overview

The following diagram illustrates the overall solution presented in this draft. Note that some of the interactions illustrated below occur at different times, only the numbered interactions (1-3) occur at the time a device is bootstrapping itself.
The boxes in this diagram are described next. A sequence diagram explaining the various calls follows in Section 2.2.

- **Vendor**

  Vendors manufacture the devices supporting NETCONF ZeroTouch. To support this solution, Vendors must support a one-time enrollment process per business organization owning the NMS. Vendors must also support sending additional information to the business organization about the devices that have been shipped for device orders it places.

- **Device**

  The devices supporting NETCONF ZeroTouch will only attempt the bootstrapping process when booting with its factory default...
configuration. As illustrated above, the bootstrapping process consists of three interactions:

1. When joining the network, the device will attempt to configure IP networking from a DHCP server. If the device is able to reach a DHCP server, it may discover additional bootstrapping information. The additional bootstrapping information consists of one or more additional Bootstrap Servers the device should try to connect to.

2. The device sequentially processes its list of Bootstrap Servers, prioritizing any that might have been learned from the DHCP server. Once the device has successfully configured itself using the bootstrapping information, it notifies the bootstrapping server for monitoring purposes.

3. Assuming the bootstrapping information configures the device appropriately, the device will initiate a NETCONF Call Home connection [draft-ietf-netconf-call-home].

More information about Devices is in Section 4.

- DHCP Server

This draft assumes the use of a DHCP server but, in reality, the solution is not intrinsically tied to using a DHCP server. Any mechanism or combination of mechanisms that can provide dynamic networking assignment would equally do.

Assuming the use of DHCP, this draft defines a specific DHCP Option for the discovering of addition bootstrapping information. More information about the ZeroTouch DHCP Option is in Section 7.1.

- Bootstrap Server

Bootstrap Servers host the bootstrapping information staged by NMSs for the devices to find. The Bootstrap Server presents a simple REST interface for devices to obtain both their bootstrapping information as well as notify the Bootstrapping Server when it has successfully completed the bootstrapping process.

Bootstrap Servers may be deployed on the public Internet or on a local network. Devices may be preconfigured with a list of well-known Bootstrap Servers. Additional Bootstrap Servers (i.e. not in the device’s preconfigured list) must be discovered from a DHCP server.
How Bootstrap Servers are deployed is out of scope of this draft, but there are a couple points worth noting. Firstly, it is expected that Internet based Bootstrap Servers will initially be hosted by Vendors, whilst waiting for 3rd-party servers to become available. Secondly, it is expected that locally administrated networks with in-house solutions might bundle the Bootstrap Server into another system (e.g., the NMS), where having the features integrated can streamline various workflows.

More information about Bootstrap Servers is in Section 3.

- **Network Management System**

  The NMS is a term used here loosely to represent any system, or collection of systems, deployed by a business organization to manage its devices. An NMS being able to establish a secure NETCONF connection with devices purchased by its organization is the ultimate goal of this solution presented by this draft. More information about the Network Management System is in Section 5.

2.2. Interactions

The following diagram illustrates the interactions between the entities described in the previous section. Note that the interactions can be roughly categorized as those that occur before a device powers on and those that occur after a device powers on.
1. imports trust anchor
2. signs up for owner cert
3. orders devices
4. ships
5. provides serial-numbers and/or IDevID cert(s), and ownership vouchers
6. stage bootstrap data
7. stage bootstrap info (optional)
8. power on
9. get networking settings and staged bootstrap info (if any)
10. update boot-image, if needed, and install config, if valid
11. netconf call-home
These interactions are described below.

1. An organization, upon deciding to deploy a Vendor’s devices for NETCONF ZeroTouch, would import into its NMS the IDevID trust anchor certificate from the Vendor. This certificate is later used by the NMS to authenticate device identities during NETCONF call home connections.

2. An organization needs to sign up to a Vendor-provided ZeroTouch program. This program entails the Vendor providing a signed Owner certificate to the organization (depicted here), as well as a commitment to sign Ownership Vouchers for future device orders (interaction #5).

3. Subsequently, the organization may place orders to the Vendor for devices supporting ZeroTouch. The ordering process may entail an explicit request for ZeroTouch support, as the Vendor providing the files in step #5 may not be enabled by default.

4. The Vendor ships the devices to the various addresses specified in the device order. For example, to an organization’s inventory warehouse, where the devices are stored in batches to supply internal requests. In another example, the devices may be shipped to their final deployment destinations.

5. In order to support ZeroTouch, the Vendor sends to the organization information about the devices it shipped. This information may be sent to the organization via email or a portal site. The information includes the serial-number and/or IDevID certificate, for each device, as well as one more Ownership Vouchers, assigning ownership for the devices to the organization.

6. In anticipation for the devices performing the ZeroTouch process, the NMS configures the Bootstrap Server. This configuration includes everything a device needs to securely connect to the NMS.

7. For deployments where the DHCP server can be customized, the NMS may configure the DHCP server to provide the device a list of additional Bootstrap Servers to consider, in addition to those the device knows of by default. This customization can be configured at a global level in the DHCP server, as it is not dependent on the type of device in any way.

8. At some point, the device powers on and, when having its factory default configuration, initiates the ZeroTouch process.
9. The device obtains from the DHCP server a dynamic network assignment. The device may also at this time discover a list of additional bootstrap servers, as optionally configured by the NMS in step #7.

10. The device iterates over its list of Bootstrap Servers, until it can successfully bootstrap its initial configuration. If it is unable to bootstrap an initial configuration, the device boots as normal. If the staged information directs the device to load a new image, it does so and reboots. If the device reboots, it continues to have a factory default configuration state, which then bring it back to this state, when it would then have the correct image. The device then loads the staged configuration into its running datastore, after validating that the configuration was signed by its rightful owner, as designated by the Ownership Voucher.

11. Assuming the bootstrapping information configures the device appropriately, the device will initiate a NETCONF Call Home [draft-ietf-netconf-call-home] connection to the NMS, which then takes over the on-going management of the device.

3. Bootstrap Server

A Bootstrap Server MUST implement the southbound interface defined below. In order to support the southbound interface, the Bootstrap Server will also need to have a northbound interface, which is described in general terms below.

3.1. Northbound Interface

The Bootstrap Server will need to provide a northbound interface of some sort to enable configuration of the bootstrapping information for the devices. Defining this interface is out of scope for this document, but it northbound interface is generally expected to:

- Enable listing, creation, modification, and deletion of entries
- Enable determining a device’s current bootstrapping state
- Enable alerting external systems when a device sends notifications

3.2. Southbound Interface

The Bootstrap Server’s southbound interface is a REST API that is described by the YANG [RFC6020] module defined in this section and presented using RESTCONF [draft-ietf-netconf-restconf]. Example usage of this API is provided in Appendix A.2.
A tree diagram describing the Bootstrap Server’s southbound interface follows:

module: ietf-zerotouch-bootstrap-server
  +--ro devices
    +--ro device* [unique-id]
      +--ro unique-id string
      +--ro ownership-voucher
        +--ro voucher binary
        +--ro issuer-crl? string
      +--ro owner-certificate
        +--ro certificate string
        +--ro issuer-crl? string
      +--ro boot-image!
        +--ro name string
        +--ro path string
        +--ro signature string
    +--ro configuration
      +--ro config
      +--ro signature string

rpcs:
  +----x notification
  +----w input
  +-----w unique-id string
  +-----w type enumeration
  +-----w message? string

In the above diagram, notice that the entire data model is read-only, as devices can only pull data from the Bootstrap Server. The data model also defines a single RPC, which is used by the device to provide asynchronous notifications.

The Bootstrap Server’s southbound interface is normatively defined by the following YANG module:

<CODE BEGINS> file "ietf-zerotouch-bootstrap-server@2015-03-09.yang"

module ietf-zerotouch-bootstrap-server {

  prefix "ztbs";

  organization "IETF NETCONF (Network Configuration) Working Group";

  contact "WG Web: <http://tools.ietf.org/wg/netconf/>

This module defines the southbound interface for ZeroTouch Bootstrap Servers.

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

revision "2015-03-09" {
  description
    "Initial version";
  reference
    "RFC XXXX: Zero Touch Provisioning for NETCONF Call Home";
}

// top-level container
container devices {
  config false;
  description
    "A list of device entries";
  list device {
    key unique-id;
    leaf unique-id { 
      type string;
    }
  }

  container ownership-voucher {
    description
      "This container contains the Ownership Voucher that the
device uses to ascertain the identity of its rightful owner, as certified by its Vendor."

leaf voucher {
    type binary;
    mandatory true;
    description
        "A Vendor-specific encoding binding unique device identifiers to an owner identifier value matching the value encoded in the owner-certificate below. An example format for a voucher is presented in the Appendix of RFC XXXX."
    }

leaf issuer-crl {
    type string;
    description
        "An absolute path to a CRL for the issuer used by the Vendor to sign Ownership Vouchers. The CRL should be as up to date as possible. This leaf is optional as it is primarily to support deployments where the device is unable to download the CRL from the CRL distribution point URLs listed in the Vendor’s trust anchor certificate."
    }

contAINER owner-certificate {
    description
        "It is intended that the device will fetch this container as a whole, as it contains values that need to be processed together."
    leaf certificate {
        type string;
        mandatory true;
        description
            "This is an X.509 certificate, signed by a Vendor, for a business organization. This certificate must encode a Vendor-assigned value identifying the organization. This identifier must match the owner identifier encoded in the Ownership Voucher."
    }
    leaf issuer-crl {
        type string;
        description
            "An absolute path to a CRL for the issuer used by the Vendor to sign Owner Certificates. The CRL should be as up to date as possible. This leaf is optional as it is primarily to support deployments where the device is unable to download the CRL from the CRL distribution point URLs listed in the Vendor’s trust anchor certificate."
    }
}
point URLs listed in the Vendor’s trust anchor
certificate.
}
}

container boot-image {
    presence
        "Only present when boot image information has been configured";
    description
        "It is intended that the device will fetch this container
        as a whole, as it contains values that need to be
        processed together."
    leaf name {
        type string;
        mandatory true;
        description
            "The name of the image of software the device is expected
            to be running.";
    }
    leaf path {
        type string;
        mandatory true;
        description
            "An absolute path to the boot-image file hosted on this
            Bootstrap server."
    }
    leaf signature {
        type string;
        mandatory true;
        description
            "The signature over the concatenation of the previous two
            leafs using the organization’s private key."
    }
}

container configuration {
    description
        "It is intended that the device will fetch this container
        as a whole, as its contents need to be processed together."
    anyxml config {
        mandatory true;
        description
            "Any configuration data model known to the device. It may
            contain Vendor-specific and/or standards-based data models.
            An example configuration using a couple IETF-defined data
            models is presented the Appendix of RFC XXXX."
    }
    leaf signature {

type string;
mandatory true;
description
"The signature over the config leaf using the
organization’s private key."
};
}

rpc notification {
input {
leaf unique-id {
  type string;
  mandatory true;
}
leaf type {
  type enumeration {
    enum boot-image-missing {
      description
      "Indicates that the device got an error when trying to
      access the provided URL";
    }
    enum boot-image-invalid {
      description
      "Indicates that the device had a problem processing the
      boot-image file (corruption)";
    }
    enum image-name-mismatch {
      description
      "Indicates that the processed boot-image contains a name
      other than provided";
    }
    enum voucher-invalid {
      description
      "Indicates that the device had a problem processing the
      voucher (chain verification failed, revoked crl)";
    }
    enum owner-cert-invalid {
      description
      "Indicates that the device had a problem processing the
      voucher (chain verification failed, revoked crl)";
    }
    enum owner-id-mismatch {
      description
      "Indicates that the owner-id in the voucher does not
      match the one inside the owner-cert";
    }
  }
}
}
enum signature-invalid {
  description
  "Indicates that the signature could not be verified using the owner-cert";
}

enum bootstrap-complete {
  description
  "Indicates that the device successfully processed the bootstrap data. At this point, the device is running the required boot-image and configuration. A device is expected to only send this notification once, assuming it does not receive an error in the HTTP response from the Bootstrap Server.";
}

mandatory true;

leaf message {
  type string;
  description
  "A human-readable value that might provide useful information";
}

4. Device

Devices supporting ZeroTouch MUST have the preconfigured factory default state and bootstrapping logic described in the following sections.

4.1. Factory Default State
1. Devices MUST be manufactured with a list of default Bootstrap Servers. Each Bootstrap Server may be identified via a hostname or an IP address. This may be an empty list if for some reason the Vendor prefers to force its devices to have to discover Bootstrap Servers from a DHCP server.

2. Devices MUST be manufactured with a list of trust anchor certificates that can be used to authenticate Bootstrap Server connections with. To support Bootstrap Servers discovered from a DHCP server, these certificates SHOULD include public certificate authorities, such as those that are included in a web browser.

3. Devices MUST be manufactured with the trust anchor certificate for Owner certificates that the Vendors provide to business organizations when they enroll in the Vendor’s ZeroTouch program. This trust anchor certificate is later used by the device to validate the Owner certificate it downloads from the Bootstrap Server.

4. Devices MUST be manufactured with the trust anchor certificate for the device ownership vouchers that the Vendors provide to organizations when it ships out an order of ZeroTouch devices. This trust anchor certificate is later used by the device to validate the Ownership Vouchers it downloads from the Bootstrap Server.
5. Devices MUST be manufactured with an initial device identifier (IDevID), as defined in [Std-802.1AR-2009]. The IDevID is an X.509 certificate, encoding a globally unique device identifier (e.g., serial number). The device MUST also possess any intermediate certificates between the IDevID certificate and the Vendor’s IDevID trust anchor certificate. These certificates are later used by the device to identify itself when it calls home. In particular, these certificates are to be used by the device’s NETCONF server, either as its SSH host-key or its TLS server certificate. Please see NETCONF Call Home [draft-ietf-netconf-call-home] for more information.

6. Device MUST be manufactured with a private key that corresponds to the public key encoded in its IDevID certificate. This private key SHOULD be securely stored, ideally by a cryptographic processor (e.g., a TPM).

4.2. Boot Sequence

Power On

1. Running default config? --------> Boot normally
   Yes
   No
2. Able to reach DHCP server? --------> Boot normally
   Yes
   No
3. Prepend any additional Bootstrap Servers discovered
   Yes
   No
4. Able to bootstrap off any Bootstrap Server? --------> Boot normally (see next diagram for drill-down details)
   Yes
   No
5. Run with new configuration

These interactions are described next.

1. When the device powers on, it first checks to see if it is running the factory default configuration. If it is running a modified configuration, then it boots normally.
2. The device tries to obtain a dynamic network assignment from a DHCP server. If it is unable to reach a DHCP server, it boots normally.

3. If the DHCP server’s offer includes the ZeroTouch Information DHCP option defined in Section 7.1, the device prepends the specified Bootstrap Servers to its factory default list.

4. The device iterates over its list of Bootstrap Servers, as described in the next section. If it is unable to bootstrap itself off any of the servers, it boots normally.

5. If the device was able to bootstrap itself off any of the Bootstrap Servers, it runs with the new configuration merged into its running datastore.

Following are the actions performed by the device when bootstrap off a Bootstrap Server (step #4 the in previous diagram).
Connect to port 443

| v | No
1. Able to validate server certificate? ———> Exit
   | Yes | v | No
2. Able to validate ownership voucher? ———> Post notification and exit
   | Yes | v | No
3. Able to validate owner certificate? ———> Post notification and exit
   | Yes | v | No
4. Able to validate boot image info? ———> Post notification and exit
   | Yes | v | No
5. Need to install boot image? ———> Install and reboot
   | Yes | v | No
6. Able to validate configuration? ———> Post notification and exit
   | Yes | v | No
7. Merge configuration into running datastore
   | v
8. Post bootstrap complete notification and exit

These interactions are described next.

1. As part of the HTTPS connection, the device will need to authenticate the server certificate presented by the Bootstrap Server. The device authenticates the server certificate using path-validation to one of its preconfigured Bootstrap Server trust anchors. If the device is unable to authenticate the server’s certificate, it abandons this Bootstrap Server and exits.

2. The device downloads the ownership voucher from the Bootstrap Server. The device validates the voucher is signed by its Vendor, using its preconfigured trust anchor for device ownership vouchers. The device also validates that its unique identifier is listed by the voucher. If the device is unable to validate the voucher or can not find its unique identifier listed, it
posts a notification message to that effect and abandons this Bootstrap Server.

3. The device downloads the owner certificate from the Bootstrap Server. The device validates that this certificate is signed by its Vendor, using path-validation to its preconfigured trust anchor for owner certificates. The device also validates that the organization identifier is the same as listed in the ownership voucher, validated in step #2. If the device is unable to validate the certificate or the owner identifier does not match, it posts a notification message to that effect and abandons this Bootstrap Server.

4. The device tries to download the boot image information. If no boot image information is available, it skips the remainder of this step. Otherwise, the device validates the boot image information using the public key from the owner certificate obtained in step #3. If it is unable to authenticate the boot image information, it posts a notification message to that effect and abandons this Bootstrap Server.

5. The device checks if the specified boot-image name matches what the device is currently running. If there is a mismatch, device downloads the new image from the Bootstrap Server and installs it. It is expected that the device will reboot itself in order to activate the new image and, further, that doing so preserves its factory default state such that it will return to this same check again, but then running the correct image. If the device is unable to install the boot-image, it posts a notification message to that effect and abandons this Bootstrap Server.

6. The device downloads the configuration from the Bootstrap Server and validates the configuration using the public key from the owner certificate obtained in step #3. If it is unable to authenticate the configuration, it posts a notification message to that effect and abandons this Bootstrap Server.

7. The device merges the configuration into its running datastore. It is expected that this configuration will provide the information necessary for the device to establish a secure NETCONF connection to its NMS using NETCONF Call Home ([draft-ietf-netconf-call-home]).

8. The device posts a bootstrap completion notification message to the Bootstrap Server and exits.
5. Network Management System (NMS)

5.1. Overview

It is expected that the bootstrapping configuration will guide the device to initiate a secure NETCONF connection to the NMS using NETCONF Call Home [draft-ietf-netconf-call-home]. This section describes what the NMS needs to do to ensure security for the device's connection.

5.2. Precondition

1. In order to authenticate a device, the NMS MUST possess the IDevID trust anchor provided by its Vendor to enable verification of the device's IDevID certificate. Specifically, the NMS uses this certificate to validate the identity certificate a device presents when negotiating the SSH or TLS transport for the NETCONF Call Home connection [draft-ietf-netconf-call-home]. Because an NMS may interoperate with multiple vendors, and a vendor may have more than one trust anchor for signing its devices IDevID certificates, this generalizes into the NMS needing a list of trust anchor certificates. These certificates SHOULD be stored in a way that prevents tampering, which is why they are shown in read-only storage in the diagram.
2. In order for the NMS to validate that a specific device connecting to it is legitimate, it MUST have a list of expected unique device identifiers (e.g., serial-numbers). The unique identifier encoded into the device’s IDevID certificate MUST match one of the expected identifiers in order for a device to be considered legitimate.

3. The NMS must have login credentials for each device. These credentials may be, for instance, a private key used for SSH or TLS client authentication. It is expected that a device is able to authenticate the NMS’s credentials by virtue of the configuration it downloads from the Bootstrap Server. These private-keys SHOULD be stored securely, such that they can not be easily compromised.

5.3. Connection Handling

When receiving a NETCONF call home connection from a device, the NSM completes the connection as specified NETCONF Call Home [draft-ietf-netconf-call-home].

6. Security Considerations

6.1. Entropy loss over time

Section 7.2.7.2 of the IEEE Std 802.1AR-2009 standard says that IDevID certificate should never expire (i.e. having a notAfter 99991231235959Z). Given the long-lived nature of these certificates, it is paramount to use a strong key length (e.g., 512-bit ECC). Vendors SHOULD deploy Online Certificate State Protocol (OCSP) responders or CRL Distribution Points (CDP) to revoke certificates in case necessary.

6.2. Serial Numbers

This draft suggests using the device’s serial number as the unique identifier in its IDevID certificate. This is because serial numbers are ubiquitous and prominently contained in invoices and on labels affixed to devices and their packaging. That said, serial numbers many times encode revealing information, such as the device’s model number, manufacture date, and/or sequence number. Knowledge of this information may provide an adversary with details needed to launch an attack.
7. IANA Considerations

7.1. ZeroTouch Information DHCP Options

The following registrations are in accordance to RFC 2939 for "BOOTP Vendor Extensions and DHCP Options" registry maintained at http://www.iana.org/assignments/bootp-dhcp-parameters.

7.1.1. DHCP v4 Option

Tag: XXX

Name: Zero Touch Information

Description: Returns a list of null-terminated Configuration Server hostnames and/or IP addresses.

<table>
<thead>
<tr>
<th>Code</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXX</td>
<td>n</td>
</tr>
</tbody>
</table>

Reference: RFC XXXX

7.1.2. DHCP v6 Option

Tag: YYY

Name: Zero Touch Information

Description: Returns a list of null-terminated Configuration Server hostnames and/or IP addresses.

<table>
<thead>
<tr>
<th>Code</th>
<th>Len</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYY</td>
<td>n</td>
</tr>
</tbody>
</table>

Reference: RFC XXXX

8. Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): David Harrington, Dean Bogdanovic, Martin Bjorklund, Stephen Hanna, Wes Hardaker, Russ Mundy, Reinaldo Penno, Randy Presuhn, Juergen Schoenwaelder.
Special thanks goes to Steve Hanna, Russ Mundy, and Wes Hardaker for brainstorming the original I-D’s solution during the IETF 87 meeting in Berlin.

9. Normative References


Appendix A. Examples

A.1. Ownership Voucher

Following describes an example data-model for an Ownership Voucher. Real vouchers are expected to be encoded in a Vendor-specific format outside the scope for this draft.

A tree diagram describing an Ownership Voucher:

```
module: ietf-zerotouch-ownership-voucher
  +--rw voucher
    +--rw unique-id* string
    +--rw owner-id string
    +--rw created-on yang:date-and-time
    +--rw expires-on? yang:date-and-time
    +--rw signature string
```

The YANG module for this example voucher:

```
<CODE BEGINS> file "ietf-zerotouch-ownership-voucher@2015-03-09.yang"

module ietf-zerotouch-ownership-voucher {
  prefix "ztov";

  import ietf-yang-types { prefix yang; }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web:  <http://tools.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    WG Chair: Mehmet Ersue
      <mailto:mehmet.ersue@nsn.com>
    WG Chair: Mahesh Jethanandani
      <mailto:mjethanandani@gmail.com>
    Editor:   Kent Watsen
      <mailto:kwatsen@juniper.net>"

  description
    "This module defines the format for a ZeroTouch ownership voucher, which is produced by Vendors, relayed by Bootstrap Servers, and consumed by devices. The purpose of the voucher is to enable a device to ascertain the identity of its rightful owner, as

certified by its Vendor.

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.

revision "2015-03-09" {
  description
    "Initial version";
  reference
    "RFC XXXX: Zero Touch Provisioning for NETCONF Call Home";
}

// top-level container
container voucher {
  leaf-list unique-id {
    type string;
    min-elements 1;
    description
      "The unique identifier (e.g., serial-number) for a device. The value must match the value in the device’s IDevID certificate. A device uses this value to determine if the voucher applies to it.";
  }
  leaf owner-id {
    type string;
    mandatory true;
    description
      "A Vendor-assigned value for the rightful owner of the devices enumerated by this voucher. The owner-id value must match the value in the owner-certificate below";
  }
  leaf created-on {
    type yang:date-and-time;
    mandatory true;
    description
      "The date this voucher was created";
  }
  leaf expires-on {
    type yang:date-and-time;
    mandatory true;
    description
      "The date this voucher expires";
  }
}

type yang:date-and-time;

description
"The date this voucher expires, if any";
}
leaf signature {

type string;

mandatory true;

description
"The signature over the concatenation of all the previous values";
}
}
}<CODE ENDS>

A.2. Bootstrap Server’s API

[‘\’ line wrapping added for formatting only]

GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server:\devices/device=123456/ownership-voucher

GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server:\devices/device=123456/owner-certificate

GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server:\devices/device=123456/boot-image

GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server:\devices/device=123456/configuration

POST https://example.com/restconf/operations/ietf-zerotouch-bootstrap-\server:notification

A.3. Bootstrap Configuration

This example illustrates a configuration enabling secure NETCONF call-home using standards-based YANG modules.
<?xml version="1.0"?>
<configuration>
  <!-- from ietf-system.yang -->
  <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
    <authentication>
      <user>
        <name>admin</name>
        <ssh-key>
          <name>admin’s rsa ssh host-key</name>
          <algorithm>ssh-rsa</algorithm>
          <key-data>AAAAB3NzaC1yc2EAAAADQABAAABABQDeJMV8zrtsi8CgEsRCjCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/O2MwjE11G9YxLzeS5p2ngzK61vikUSqfMukeBohFTFrDZ8bUrF+HMLlTRncCVCCWAwl1Or9IDGDAXuw6G45gLcHalHMMbTQxKn2dzU9kxf/13Z5S5G76Py6sA5vg7SilqQPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2KqBIe340jWqEJua7LtvEYJq4wnq44Iog/*CiumTkmQIWRgIoJ4FCzYbO9nVEf4O1f6gakWVOZZgQ8929uWjCWI1Glqn2mPibp2Go1</key-data>
        </ssh-key>
      </user>
    </authentication>
  </system>

  <!-- from ietf-netconf-server.yang -->
  <netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
    <call-home>
      <application>
        <name>config-mgr</name>
        <ssh>
          <endpoints>
            <endpoint>
              <name>east-data-center</name>
              <address>11.22.33.44</address>
            </endpoint>
            <endpoint>
              <name>west-data-center</name>
              <address>55.66.77.88</address>
            </endpoint>
          </endpoints>
        </ssh>
      </application>
    </call-home>
  </netconf-server>
</configuration>
Appendix B. Change Log

B.1. ID to 00

- Major structural update; the essence is the same. Most every section was rewritten to some degree.
- Added a Use Cases section
- Added diagrams for "Actors and Roles" and "NMS Precondition" sections, and greatly improved the "Device Boot Sequence" diagram
- Removed support for physical presence or any ability for Configlets to not be signed.
- Defined the ZeroTouch Information DHCP option
- Added an ability for devices to also download images from Configuration Servers
- Added an ability for Configlets to be encrypted
- Now Configuration Servers only have to support HTTP/S - no other schemes possible

B.2. 00 to 01

- Added boot-image and validate-owner annotations to the "Actors and Roles" diagram.
- Fixed 2nd paragraph in section 7.1 to reflect current use of anyxml.
- Added encrypted and signed-encrypted examples
- Replaced YANG module with XSD schema
- Added IANA request for the ZeroTouch Information DHCP Option
- Added IANA request for media types for boot-image and configuration

B.3. 01 to 02

- Replaced the need for a Configuration Signer with the ability for each NMS to be able to sign its own configurations, using Vendor signed Ownership Vouchers and Owner certificates.
Renamed Configuration Server to Bootstrap Server, a more representative name given the information devices download from it.

Replaced the concept of a Configlet by defining a southbound interface for the Bootstrap Server using YANG.

Removed the IANA request for the boot-image and configuration media types

Authors’ Addresses
Kent Watsen
Juniper Networks
EMail: kwatsen@juniper.net

Joe Clarke
Cisco Systems
EMail: jclarke@cisco.com

Mikael Abrahamsson
T-Systems
EMail: "mikael.abrahamsson@t-systems.se"
Abstract

This document specifies automated bootstrapping of a key infrastructure using vendor installed IEEE 802.1AR manufacturing installed certificates, in combination with a vendor based service on the Internet. Before being authenticated, a new device has only link-local connectivity, and does not require a routable address. When a vendor provides an Internet based service, devices can be forced to join only specific domains but for constrained environments we describe a variety of options that allow bootstrapping to proceed.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 17, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction ................................................. 3
   1.1. Terminology ............................................. 4
2. Architectural Overview ....................................... 4
3. Operational Overview ......................................... 7
   3.1. Instantiating the Domain Certification Authority ......... 7
   3.2. Instantiating the Registrar .............................. 7
   3.3. Accepting New Entities ................................. 8
   3.4. Automatic Enrolment of Devices .......................... 9
   3.5. Operating the Network ................................... 9
4. Functional Overview ........................................... 9
   4.1. Behavior of a new entity ................................ 10
   4.1.1. Proxy Discovery ..................................... 11
   4.1.2. Receiving and accepting the Domain Identity .......... 12
   4.1.3. Enrollment .......................................... 13
   4.1.4. After Enrollment .................................... 13
   4.2. Behavior of a proxy .................................... 13
   4.3. Behavior of the Registrar ............................... 14
   4.3.1. Authenticating the Device ............................ 14
   4.3.2. Accepting the Entity ................................ 14
   4.3.3. Claiming the new entity .............................. 15
   4.4. Behavior of the MASA Service ........................... 15
   4.4.1. Issue Authorization Token and Log the event .......... 16
   4.4.2. Retrieve Audit Entries from Log ...................... 16
   4.5. Leveraging the new key infrastructure / next steps ..... 16
   4.5.1. Network boundaries ................................. 16
5. Protocol Details .............................................. 17
   5.1. EAP-EST ................................................ 18
   5.2. Request bootstrap token ................................ 18
   5.3. Request MASA authorization token ....................... 18
   5.4. Request MASA authorization log ........................ 19
6. Reduced security operational modes ............................ 20
7. Security Considerations ...................................... 21
   7.1. Trust Model ............................................ 21
8. Acknowledgements ............................................. 22
9. References .................................................. 22
   9.1. Normative References ................................... 22
   9.2. Informative References ................................ 22
Authors’ Addresses ............................................. 23
1. Introduction

To literally "pull yourself up by the bootstraps" is an impossible action. Similarly the secure establishment of a key infrastructure without external help is also an impossibility. Today it is accepted that the initial connections between nodes are insecure, until key distribution is complete, or that domain-specific keying material is pre-provisioned on each new device in a costly and non-scalable manner. This document describes a zero-touch approach to bootstrapping an entity by securing the initial distribution of key material using third-party generic keying material, such as a manufacturer installed IEEE 802.1AR certificate [IDevID], and a corresponding third-party service on the Internet.

The two sides of an association being bootstrapped authenticate each other and then determine appropriate authorization. This process is described as four distinct steps between the existing domain and the new entity being added:

- New entity authentication: "Who is this? What is its identity?"
- New entity authorization: "Is it mine? Do I want it? What are the chances it has been compromised?"
- Domain authentication: "What is this domain’s claimed identity?"
- Domain authorization: "Should I join it?"

A precise answer to these questions can not be obtained without leveraging an established key infrastructure(s). The domain’s decisions are based on the new entity’s authenticated identity, as established by verification of previously installed credentials such as a manufacturer installed IEEE 802.1AR certificate, and verified back-end information such as a configured list of purchased devices or communication with a trusted third-party. The new entity’s decisions are made according to verified communication with a trusted third-party or in a strictly auditable fasion.

Optimal security is achieved with IEEE 802.1AR certificates on each new entity, accompanied by a third-party Internet based service for verification. The concept also works with less requirements, but is then less secure. A domain can choose to accept lower levels of security when a trusted third-party is not available so that bootstrapping proceeds even at the risk of reduced security. Only the domain can make these decisions based on administrative input and known behavior of the new entity.
The result of bootstrapping is that a domain specific key infrastructure is deployed. Since IEEE 802.1AR PKI certificates are used for identifying the new entity and the public key of the domain identity is leveraged during communications with an Internet based service, which is itself authenticated using HTTPS, bootstrapping of a domain specific Public Key Infrastructure (PKI) is fully described. Sufficient agility to support bootstrapping alternative key infrastructures (such as symmetric key solutions) is considered although no such key infrastructure is described.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms are defined for clarity:

2. Architectural Overview

The logical elements of the bootstrapping framework are described in this section. Figure 1 provides a simplified overview of the components. Each component is logical and may be combined with other components as necessary.
Figure 1

Domain: The set of entities that trust a common key infrastructure trust anchor.

Domain CA: The domain Certification Authority (CA) provides certification functionalities to the domain. At a minimum it provides certification functionalities to the Registrar and stores...
the trust anchor that defines the domain. Optionally, it certifies all elements.

Domain Identity: The domain identity is the 160-bit SHA-1 hash of the BIT STRING of the subjectPublicKey of the domain trust anchor that is stored by the Domain CA. This is consistent with the RFC5280 Certification Authority subject key identifier of the Domain CA’s self signed root certificate. (A string value bound to the Domain CA’s self signed root certificate subject and issuer fields is often colloquially used as a humanized identity value but during protocol discussions the more exact term as defined here is used).

Orchestrator: Although bootstrapping of an individual device is automated and requires zero administrative involvement (particularly on the New Entity) the orchestrator drives general operations of the domain. This can be an automated process or a human administrator, see Section 3.3 for more details.

Factory: This instantiates the New Entity. For physical devices this can be representative of third-party vendor manufacturing, ordering and shipping process(es) that results in a physical hardware device with an IEEE 802.1AR identity being drop shipped to a destination domain for physical installation. In a virtual machine environment this can be the virtual machine hypervisor control software that initiates a virtual machine instance, in which case the factory is a "virtual factory" and might be managed by the domain itself.

Factory CA: This Certification Authority is leveraged by the Factory to issue IEEE 802.1AR identities to each New Entity. For a virtual factory it may be reasonable to assume the domain certification authority is directly used but in a complex environment it is assumed the Factory does not have direct access to the Domain Certification Authority.

Registrar: A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a Registrar to control this process. Typically a Registrar is "inside" its domain.

New Entity: A new device or virtual machine or software component that is not yet part of the domain.

Proxy: A domain entity that helps the New Entity join the domain. A Proxy facilitates communication for devices that find themselves
in an environment where they are not provided L3 connectivity
until after they are validated as members of the domain.

MASA Service: A Manufacturer Authorized Signing Authority (MASA)
service on the global Internet. At a minimum the MASA provides a
trusted repository for audit information concerning privacy
protected bootstrapping events. As a service offering the MASA
can incorporate many of the bootstrapping elements (such as the
Registrar and the Domain CA) into the overall service. The MASA
is not a mandatory component, but it enables the new device to
validate which domain it is joining. This allows for a completely
secure zero-touch bootstrap of domain certificates with mutual
authentication (device <-> domain).

We assume a multi-vendor network. In such an environment, there
could a MASA for each vendor that supports devices following this
document’s specification, or an integrator could provide a MASA
service for all devices which he supplies. Note again that the MASA
is not mandatory. Also, this approach describes a secure zero-touch
approach to bootstrapping a key infrastructure; if certain devices in
a network do not support this approach, they can still be
bootstrapped manually.

3. Operational Overview

This section describes how an operator interacts with a domain that
supports the bootstrapping as described in this document.

3.1. Instantiating the Domain Certification Authority

This is a one time step by the domain administrator. This is an "off
the shelf" CA with the exception that it is designed to work as an
integrated part of the security solution. This precludes the use of
3rd party certification authority services that do not provide
support for delegation of certificate issuance decisions to a domain
managed Registration Authority.

3.2. Instantiating the Registrar

This is a one time step by the domain administrator. One or more
devices in the domain are configured take on a Registrar function.

A device can be configured to act as a Registrar or a device can
auto-select itself to take on this function, using a detection
mechanism to resolve potential conflicts and setup communication with
the Domain Certification Authority. Automated Registrar selection is
outside scope for this document.
3.3. Accepting New Entities

For each New Entity the Registrar is informed the unique identifier (e.g. serial number) along with the manufacturer's identifying information (e.g. manufacturer root certificate). This can happen in different ways:

1. Default acceptance: In the simplest case, the new device itself provides its identity to the registrar, which then accepts any device blindly, without validating its identity. This mode does not provide any security against intruders and is not recommended.

2. Per device acceptance: Also here the device provides its identity directly to the registrar during enrollment. A non-technical human validates the identity, for example by comparing the identity displayed by the registrar (for example using a smartphone app) with the identity shown on the packaging of the device. Acceptance may be triggered by a click on a smartphone app "accept this device", or by other forms of pairing. See also [I-D.behringer-homenet-trust-bootstrap] for how the approach could work in a homenet.

3. Whitelist approach: In larger networks, neither of the previous approaches is acceptable. Default acceptance is not secure, and a manual real-time acceptance per device does not scale. Here, the registrar is provided a priori with a list of identifiers of devices that belong to the network. This list can be for example extracted from an inventory database, or sales records. If a device is detected that is not on the list of known devices, it can still be manually accepted or declined.

4. Automated Orchestrator: an automated process that queries the MASA service or an inventory database either a priori for all devices, or in real time for each new device. It feeds this information into the Registrar. Once set up, no human intervention is required in this process.

None of the approaches requires the network to have permanent Internet connectivity. Even when the Internet based MASA service is used, it is possible to pre-fetch the required information from the MASA a priori, for example at time of purchase. In this case devices can enrol later even in a completely isolated network.

Additional policy can be stored for future authorization decisions. For example an expected deployment time window or that a certain Proxy must be used.
3.4. Automatic Enrolment of Devices

The approach outlined in this document provides a secure zero-touch method to enrol new devices without any pre-staged configuration. New devices communicate with already enrolled devices of the domain, which proxy between the new device and a Registrar. As a result of this completely automatic operation, all devices obtain a domain based certificate.

3.5. Operating the Network

The certificate installed in the previous step can be used for all subsequent operations. For example, to determine the boundaries of the domain: If a neighbor has a certificate from the same trust anchor it can be assumed "inside" the same organization; if not, as outside. See also Section 4.5.1. The certificate can also be used to securely establish a connection between devices and central control functions. Also autonomic transactions can use the domain certificates to authenticate and/or encrypt direct interactions between devices. The usage of the domain certificates is outside scope for this document.

4. Functional Overview

Entities behave in an autonomic fashion. They discover each other and autonomically establish a key infrastructure delimiting the autonomic domain. See [I-D.irtf-nmrg-autonomic-network-definitions] for more information.

The overall flow is shown in Figure 2:
4.1. Behavior of a new entity

A New Entity that has not yet been bootstrapped attempts to find a local domain and join it. A number of methods are attempted for establishing communications with the domain in a specified order.

Client behavior is as follows:

1. Discover a communication channel to the "closest" Registrar by trying the following steps in this order:
A. Search for a Proxy on the local link using a link local
discovery protocol (no routable addresses are required for
this approach). If multiple local proxies are discovered
attempt communications with each before widening the search
to other options. The proxy relays information to the
registrar. If this fails:

B. Obtain an IP address using existing methods, such as SLAAC or
DHCPv6, and search for a local registrar using DNS service
discovery. If this fails:

C. Obtain an IP address (as above), and search for the domain
registrar using a pre-defined Factory provided Internet based
re-direct service. Various methods could be used, such as
DNS or RESTful APIs.

2. Present IEEE 802.1AR credentials to the discovered Registrar (via
a Proxy if necessary). Included is a generated nonce that is
specific to this attempt.

3. Verify the MASA service generated authorization token as provided
by the contacted Registrar. The authorization token contains the
valid domain(s) for this device and is signed by the MASA
service. The device uses a pre-installed certificate of the MASA
service to validate the signature of the MASA. The nonce
information previously provided is also checked, if it was not
removed by the Registrar.

4. If and only if step three is successful: Join Domain, by
accepting the domain specific information from the registrar, and
by enrolling a domain certificate from the registrar.

5. The New Entity is now a member of the domain and will only repeat
the discovery aspects of bootstrapping if it is returned to
factory default settings.

The following sections describe each of these steps in more detail.

4.1.1. Proxy Discovery

Existing protocols provide the appropriate functionality for both
discovering the Proxy and facilitating communication through the
Proxy:

IEEE 802.1X Where the New Entity can be cast as the "supplicant" and
the Proxy is the "authenticator". The bootstrapping protocol
messages are encapsulated as EAP methods. The "authenticator"
reencapsulates the EAPOL frames and forwards them to the "Authentication Server", which provides Registrar functionalities.

PANA [RFC5191] [[EDNOTE: TBD]]


Each provides a method for the New Entity to discover and initiate communication with a local neighbor. In each protocol methods are available to support encapsulation of the bootstrapping protocol messages described elsewhere in this document. Other protocols for transporting bootstrapping messages can be added in future references.

All security associations established are between the new device and the Registrar regardless of proxy operations.

If multiple proxies are available the New Entity tries each until a successful bootstrapping occurs. The New Entity may prioritize proxies selection order as appropriate for the anticipated environment.

If Proxy discovery fails the New Entity moves on to discovering a Registrar directly.

4.1.2. Receiving and accepting the Domain Identity

The domain trust anchor is received by the New Entity during the bootstrapping protocol exchange.

An enrollment protocol such as EST [RFC7030] details a set of non-autonomic bootstrapping methods such as:

- using the Implicit Trust Anchor database (not an autonomic solution because the URL must be securely distributed),
- engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),
- and using a Certificate-Less TLS mutual authentication method (not an autonomic solution because the distribution of symmetric key material is not autonomic).
This document describes an additional autonomic method:

MASA authorization token  Authorization tokens are obtained by the Registrar from the MASA service and presented to the New Entity for validation.

If the autonomic methods fails the New Entity returns to discovery state and attempts bootstrapping with the next available discovered Registrar.

4.1.3. Enrollment

As the final step of bootstrapping a Registrar helps to issue a domain specific credential to the New Entity. For simplicity in this document, a Registrar primarily facilitates issuing a credential by acting as an RFC5280 Registration Authority for the Domain Certification Authority.

Enrollment proceeds as described in Enrollment over Secure Transport (EST) [RFC7030]. The New Entity contacts the Registrar using EST as indicated:

- The New Entity is authenticated using the IEEE 802.1AR credentials. (EST support for .
- The EST section 4.1.3 CA Certificates Response is verified using the MASA authorization token provided domain identity.

4.1.4. After Enrollment

Functionality to provide generic "configuration" is supported. The parsing of this data and any subsequent use of the data, for example communications with a Network Management System is out of scope but is expected to occur after bootstrapping enrollment is complete.

See Section 4.5.

4.2. Behavior of a proxy

The role of the Proxy is to facilitate communications. The Proxy forwards messages between the New Entity and a Registrar. Where existing protocols, as detailed in Section 4.1.1, already provide this functionality nothing additional is defined.
4.3. Behavior of the Registrar

Once a registrar is established it listens for new entities and determines if they can join the domain. The registrar delivers any necessary authorization information to the new device and facilitates enrollment with the domain PKI.

Registrar behavior is as follows:

4.3.1. Authenticating the Device

The applicable authentication methods detailed in EST [RFC7030] are:

- the use of an IEEE 802.1AR IDevID credential,
- or the use of a secret that is transmitted out of band between the New Entity and the Registrar (this use case is not autonomic).

4.3.2. Accepting the Entity

In a fully automated network all devices must be securely identified.

A Registrar accepts or declines a request to join the domain, based on the authenticated identity presented and other policy defined criteria such as Proxy identity. Automated acceptance criteria include:

- allow any device of a specific type (as determined by the IEEE 802.1AR device identity),
- allow any device from a specific Factory (as determined by the IEEE 802.1AR identity),
- allow a specific device from a Factory (as determined by the IEEE 802.1AR identity)

In all cases a Registrar must use the globally available MASA service to verify that the device’s history log does not include unexpected Registrars. Because if a device had previously registered with another domain, the registrar of that domain would show in the log.

If a device is accepted into the domain, it is then invited to request a domain certificate through a certificate enrolment process. The result is a common trust anchor and device certificates for all autonomic devices in a domain. These certificates can subsequently be used to determine the boundaries of the homenet, to authenticate other domain nodes, and to autonomically enable services on the homenet.
For each entity that will be accepted a Registrar maintains the Factory CA identity and the entity’s unique identifier. The Factory CA identity could be implemented as the Factory CA root certificate keyIdentifier (the 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey). For user interface purposes the keyIdentifier information can be mapped to a colloquial Factory name (Registrars can be shipped with the keyIdentifier of a significant number of third-party manufacturers).

4.3.3. Claiming the new entity

During initial bootstrapping the New Entity provides a nonce specific to the particular bootstrapping attempt. The registrar should include this nonce when claiming the New Entity from the Internet based MASA service. If a nonce is provided by the Registrar, then claims from an unauthenticated Registrar are serviced by the MASA resource.

The Registrar can claim a New Entity that is not online by forming the request using the entities unique identifier but not including a nonce in the claim request. MASA authorization tokens obtained in this way do not have a lifetime and they provide a permanent method for the domain to claim the device. Evidence of such a claim is provided in the audit log entries available to any future Registrar. Such claims reduce the ability for future domains to secure bootstrapping and therefore the Registrar MUST be authenticated by the MASA service.

Claiming an entity establishes an audit log at the MASA server and provides the Registrar with proof, in the form of a MASA authorization token, that the log entry has been inserted. As indicated in Section 4.1.2 a New Entity will only proceed with bootstrapping if a validated MASA authorization token has been received. The New Entity therefore enforces that bootstrapping only occurs if the claim has been logged.

4.4. Behavior of the MASA Service

The MASA service is provided by the Factory provider on the global Internet. The URI of this service is well known. The URI should be provided as an IEEE 802.1AR IDevID X.509 extension (a "MASA authorization token Distribution Point" extension).

The MASA service provides the following functionalities to Registrars:
4.4.1. Issue Authorization Token and Log the event

A Registrar POSTs a claim message optionally containing the bootstrap nonce to the MASA server.

If a nonce is provided the MASA service responds to all requests. The MASA service verifies the Registrar is representative of the domain and generates a privacy protected log entry before responding with the authorization token.

If a nonce is not provided then the MASA service MUST authenticate the Registrar as a valid customer. This prevents denial of service attacks. The specific level of authentication provided by the customer is not defined here. An MASA Practice Statement (MPS) similar to the Certification Authority CPS, as defined in RFC5280, is provided by the Factory such that Registrar’s can determine the level of trust they have in the Factory.

4.4.2. Retrieve Audit Entries from Log

When determining if a New Entity should be accepted into a domain the Registrar retrieves a copy of the audit log from the MASA service. This contains a list of privacy protected domain identities that have previously claimed the device. Included in the list is an indication of the time the entry was made and if the nonce was included.

4.5. Leveraging the new key infrastructure / next steps

As the devices have a common trust anchor, device identity can be securely established, making it possible to automatically deploy services across the domain in a secure manner.

Examples of services:

- Device management.
- Routing authentication.
- Service discovery.

4.5.1. Network boundaries

When a device has joined the domain, it can validate the domain membership of other devices. This makes it possible to create trust boundaries where domain members have higher level of trusted than external devices. Using the autonomic User Interface, specific devices can be grouped into to sub domains and specific trust levels can be implemented between those.
5. Protocol Details

For simplicity the bootstrapping protocol is described as extensions to EST [RFC7030].

EST provides a bootstrapping mechanism for new entities that are configured with the URI of the EST server such that the Implicit TA database can be used to authenticate the EST server. Alternatively EST clients can "engage a human user to authorize the CA certificate using out-of-band data such as a CA certificate". EST does not provide a completely automated method of bootstrapping the PKI as both of these methods require some user input (either of the URI or authorizing the CA certificate).

This section details additional EST functionality that support automated bootstrapping of the public key infrastructure. These additions provide for fully automated bootstrapping. These additions are to be optionally supported by the EST server within the same .well-known URI tree as the existing EST URIs.

The "New Entity" is the EST client and the "Registrar" is the EST server.

The extensions for the client are as follows:

- The New Entity provisionally accept the EST server certificate during the TLS handshake as detailed in EST section 4.1.1 ("Bootstrap Distribution of CA Certificates").

- The New Entity request and validates a "bootstrap token" as described below. At this point the New Entity has sufficient information to validate domain credentials.

- The New Entity calls the EST defined /cacerts method to obtain the current CA certificate. These are validated using the "bootstrap token".

- The New Entity completes bootstrapping as detailed in EST section 4.1.1.

These extensions could be implemented as an independent protocol from EST but since the overlap with basic enrollment is extensive, particularly with respect to client authorization, they are presented here as additions to EST.

In order to obtain a validated bootstrap token and history logs the Registrar contacts the MASA service Service using REST calls.
5.1. EAP-EST

In order to support Proxy environments EAP-EST is defined.

[[EDNOTE: TBD. EST is TLS with some data. EAP-TLS and other similar protocols provide an example framework for filling out this section]]

5.2. Request bootstrap token

When the New Entity reaches the EST section 4.1.1 "Bootstrap Distribution of CA Certificates" state but wishes to proceed in a fully automated fashion it makes a request for a MASA authorization token from the Registrar.

This is done with an HTTPS POST using the operation path value of "/requestbootstraptoken".

The request format is JSON object containing a nonce.

Request media type: application/masanonce

Request format: a json file with the following:

{"nonce":"<64bit nonce value>"}

[[EDNOTE: exact format TBD. There is an advantage to having the client sign the nonce (similar to a PKI Certification Signing Request) since this allows the MASA service to confirm the actual device identity. It is not clear that there is a security benefit from this.]]

The Registrar validates the client identity as described in EST [RFC7030] section 3.3.2. The registrar performs authorization as detailed in Section 4.3.2. If authorization is successful the Registrar obtains a MASA authorization token from the MASA service (see Section 5.3).

The received MASA authorization token is returned to the New Entity.

5.3. Request MASA authorization token

A registrar requests the MASA authorization token from the MASA service using a REST interface.

This is done with an HTTP POST using the operation path value of "/requestMASAauthorization".
The request format is a JSON object optionally containing the nonce value (as obtained from the bootstrap request) and the IEEE 802.1AR identity of the device as a serial number (the full certificate is not needed and no proof-of-possession information for the device identity is included). The New Entity’s serial number is extracted from the subject name:

```
{"nonce":"<64bit nonce value>", "serialnumber": "<subjectname/subjectaltname serial number>"}
```

Inclusion of the nonce is optional because the Registrar might request an authorization token when the New Entity is not online, or when the target bootstrapping environment is not on the same network as the MASA server.

This information is encapsulated in a PKCS7 signed data structure that is signed by the Registrar. The entire certificate chain, up to and including the Domain CA, is included in the PKCS7.

The MASA service checks the internal consistency of the PKCS7 but is unable to actually authenticate the domain identity information. The domain is not know to the MASA server in advance and a shared trust anchor is not implied. The MASA server verifies that the PKCS7 is signed by a Registrar (by checking for the cmc-idRA field in the Registrar certificate) certificate that was issued by the root certificate included in the PKCS7.

The domain ID is extracted from the root certificate and is used to generate the MASA authorization token and to update the audit log.

[[EDNOTE: This assumes the Registrar can extract the serial number successfully from the client certificate. The RFC4108 hardwareModuleName is likely the best known location.]]

5.4. Request MASA authorization log

A registrar requests the MASA authorization log from the MASA service using this EST extension.

This is done with an HTTP GET using the operation path value of "/requestMASAlog".

The log data returned is a file consisting of all previous log entries. For example:
Distribution of a large log is less than ideal. This structure can be optimized as follows: only the most recent nonce’d log entry is required in the response. All nonce-less entries for the same domainID can be condensed into the single most recent nonceless entry.

The Registrar uses this log information to make an informed decision regarding the continued bootstrapping of the New Entity.

[[EDNOTE: certificate transparency might offer an alternative log entry method]]

6. Reduced security operational modes

A common requirement of bootstrapping infrastructures is often that they support less secure operational modes. To support these operational modes the Registrar can choose to accept devices using less secure methods. For example:

1. The registrar may choose to accept all devices, or all devices of a particular type, at the administrator’s discretion. This may occur when: Informing the Registrar of unique identifiers of new entities might be operationally difficult.

2. The registrar may choose to accept devices that claim a unique identity without the benefit of authenticating that claimed identity. This may occur when: The New Entity does not include an IEEE 802.1AR factory installed credential.

3. A representative of the Registrar (e.g. the Orchestrator) may request nonce-less authorization tokens from the MASA service when network connectivity is available. These tokens can then be transmitted to the Registrar and stored until they are needed during bootstrapping operations. This may occur when: The target...
network is protected by an air gap and therefore can not contact the MASA service during New Entity deployment.

4. The device may have an operational mode where it skips authorization token validation. For example if a physical button is depressed during the bootstrapping operation. This may occur when: A device Factory goes out of business or otherwise fails to provide a reliable MASA service.

5. The device may not require the MASA service authorization token. An entity that does not validate the domain identity is inherently dangerous as it may contain malware. This risk should be mitigated using attestation and measurement technologies. In order to support an unsecured imprint the New Entity MUST support remote attestation technologies such as is defined by the Trusted Computing Group. ([EDNOTE: How to include remote attestation into the bootstrapping protocol exchange is TBD]). This may occur when: The device Factory does not provide a MASA service.

7. Security Considerations

In order to support a variety of use cases, devices can be claimed by a registrar without proving possession of the device in question. This would result in a nonceless, and thus always valid, claim. The MASA service is required to authenticate such Registrars but no programmatic method is provided to ensure good behavior by the MASA service. Nonceless entries into the audit log therefore permanently reduce the value of a device because future Registrars, during future bootstrap attempts, must now be configured with policy to ignore previously (and potentially unknown) domains.

Future registrars are recommended to take the audit history of a device into account when deciding to join such devices into their network.

It is possible for an attacker to send an authorization request to the MASA service directly after the real Registrar obtains an authorization log. If the attacker could also force the bootstrapping protocol to reset there is a theoretical opportunity for the attacker to use the authorization token to take control of the New Entity but then proceed to enrol with the target domain. To prevent this the MASA service is rate limited to only generate authorization tokens at a rate of 1 per minute. The Registrar therefore has at least 1 minute to get the response back to the New Entity. ([EDNOTE: a better solution can likely be found. This text captures the issue for now.]) Also the Registrar can double check the log information after enrolling the New Entity.
The MASA service could lock a claim and refuse to issue a new token. Or the MASA service could go offline (for example if a vendor went out of business). This functionality provides benefits such as theft resistance, but it also implies an operational risk. This can be mitigated by Registrars that request nonce-less authorization tokens.

7.1. Trust Model

[[EDNOTE: (need to describe that we need to trust the device h/w. To be completed.]]}

8. Acknowledgements

We would like to thank the various reviewers for their input, in particular Markus Stenberg, Michael Richardson, Brian Carpenter, Fuyu Eleven.

9. References

9.1. Normative References


9.2. Informative References


Authors’ Addresses

Max Pritikin
Cisco
Email: pritikin@cisco.com

Michael H. Behringer
Cisco
Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco
Email: sbjarnas@cisco.com
Abstract

This document describes the Autonomic Distributed Node Consensus Protocol (ADNCP), a profile of Distributed Node Consensus Protocol (DNCP) for autonomic networking.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

DNCP [I-D.ietf-homenet-dncp] provides a single-area link state database for arbitrary use. ADNCP extends DNCP in several ways and makes it implementable by defining a profile.

ADNCP allows for several types of point-to-point exchanges that match typical autonomic operations. The shared state within ADNCP itself is used to also facilitate some autonomic operations. Whether point-to-point or multi-party algorithms are used is left up to the specification of particular objectives.

To provide for better scalability than the base DNCP, ADNCP also defines (optionally zero-configuration) multi-area system.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].
3. Terminology

Reader is assumed to be familiar with the autonomic networking terminology described in [I-D.irtf-nmrg-autonomic-network-definitions] and [I-D.ietf-homenet-dncp].

(ADNCP) area: A set of ADNCP running nodes that are directly connected using a set of DNCP connections. In other words, DNCP network. They share a link state database, and may also have some other data from other areas but no actual topology of the other areas.

(ADNCP) network: A set of connected ADNCP areas.

area owner: The ADNCP node with the highest Node Identifier within the ADNCP area.

connection owner: Either ADNCP node with the highest Node Identifier on a multicast-capable link the connection maps to, or the unicast "server" node that other nodes connect.

per-area: Applicable to the nodes in a particular area.

area-wide: Distribution scope in which content is made available to nodes in only one area.

per-net: Applies to the whole (ADNCP) network.

net-wide: Distribution scope in which content is made available to nodes in all areas.

4. DNCP Profile

ADNCP is defined as a profile of DNCP [I-D.ietf-homenet-dncp] with the following parameters:

- ADNCP uses UDP datagrams on port ADNCP-UDP-PORT as a multicast transport over IPv6 using group All-ADNCP-Nodes-6, or IPv4 using group All-ADNCP-Nodes-4. TLS [RFC5246] on port ADNCP-TCP-PORT is used for unicast transport. Non-secure unicast transport MUST NOT be used and therefore is not defined at all. In a typical case, multicast transport SHOULD be link-local scoped, although other scopes MAY be also used and supported if multicast routing is available.
o ADNCP operates over either unicast connections, or over multicast-capable interfaces. Therefore the value encoded in the DNCP Connection Identifier is left up to the implementation.

o ADNCP nodes MUST support the X.509 PKI-based trust method, and MAY support the DNCP Certificate Based Trust Consensus method.

o ADNCP nodes MUST use the leading 128 bits of SHA256 [RFC6234] as DNCP non-cryptographic hash function H(x).

o ADNCP uses 128-bit node identifiers (DNCP_NODE_IDENTIFIER_LENGTH = 128). A node implementing ADNCP MUST generate their node identifier by applying the SHA256 to their public key. If the node receives a Node State TLV with the same node identifier and a higher update sequence number multiple times, an error SHOULD be made visible to an administrator.

o ADNCP nodes MUST NOT send multicast Long Network State messages, and received ones MUST be ignored.

o ADNCP nodes use the following Trickle parameters:
  * k SHOULD be 1, given the timer reset on data updates and retransmissions should handle packet loss.
  * Imin SHOULD be 200 milliseconds but SHOULD NOT be lower. Note: Earliest transmissions may occur at Imin / 2.
  * Imax SHOULD be 7 doublings of Imin (i.e. 25.6 seconds) but SHOULD NOT be lower.

o ADNCP nodes MUST use the keep-alive extension on all multicast interface-based connections. The default keep-alive interval (DNCP_KEEPALIVE_INTERVAL) is 20 seconds, the multiplier (DNCP_KEEPALIVE_MULTIPLIER) MUST be 2.1, the grace-interval (DNCP_GRACE_INTERVAL) SHOULD be equal to DNCP_KEEPALIVE_MULTIPLIER times DNCP_KEEPALIVE_INTERVAL.

5. Point-To-Point Operations

For point-to-point operations such as discovery, negotiation, and synchronization, a single new class of DNCP messages is defined (TBD - more detail?). It is identified by the presence of an objective-specific TLV, and if specified by the objective, it SHOULD be responded to only via unicast at most. Therefore, if an ADNCP implementation does not recognize a message, it MUST be silently ignored. These messages SHOULD NOT in and of themselves establish a
6. Distributed Operations

6.1. Discovery

If point-to-point discovery (using either multicast-capable interface(s), or known unicast peers) is not chosen, discovery can be handled also either by participating in the ADNCP network, or by performing point-to-point operation with a node participating in the ADNCP.

Presence (or lack) of content with ADNCP can be used to discover nodes that support particular objectives in some specific way; for example, an objective might specify TLV which contains an address of some particular type of server (for example, DHCPv6 PD), and therefore by just using ADNCP information, "closest" node (in terms of areas / in terms of routing of the address) could be determined.

6.2. Negotiation / Synchronization

ADNCP is not suitable for (especially net-wide) transmission of any data that changes rapidly. Therefore it should be used to sparingly publish data that changes at most gradually.

With that limitation in mind, ADNCP can be used to implement arbitrary multi-party algorithms, such as Prefix Assignment [I-D.ietf-homenet-prefix-assignment]. Given appropriate per-area hierarchical assignment (published net-wide), it could be also employed net-wide though, as the per-net prefix assignments would change only rarely.

For rapidly changing data, point-to-point exchanges (as needed) should be used instead and just e.g. relevant IP addresses published via ADNCP.

6.3. Intent Distribution

Arbitrary (operator-supplied) objective-specific intent can be supplied as TLVs within ADNCP, either per-area or per-network.
7. Area Support

Area support for DNCP is added so that non-area-capable implementations can benefit from it, but cannot support more than one interface (for same DNCP instance at any rate), as they cannot handle the logic for transferring data between areas.

Areas are uniquely identified by a 32-bit Area Identifier.

7.1. Area Boundaries

A single connection always belongs to exactly one area. Therefore, the boundaries of the areas are within nodes that have multiple connections, and can transfer data between them.

For every remote area detected (=on other connections, not on that particular connection), a node should include a Remote Area TLV which contains an Area Identifier, a Node Identifier of the area owner, and a pared down (recursive) list of Remote Area TLVs from that area, that MUST be loop free. An exception to the rule is the current area; if the current area is advertised elsewhere, it MUST be included if and only if the owner’s Node Identifier differs from the local one. Longer paths to particular areas with matching owner Node Identifier MAY be also omitted.

TBD: Remote Area TLV - area id, area owner (+container for more Remote Area TLVs recursively)

7.2. Area Identifier

Area Identifier for every connection is chosen by the connection owner. The link is owned by the node with the highest Node Identifier on a connection which consists of a multicast-capable link, or the "server" node which other nodes are connecting to in case of an unicast link.

TBD: Area Identifier TLV - just area id - originated by the area owner, and then included in every unicast message on link.

7.3. Area Formation

Areas by definition are connected parts of the network. An operator may set explicit values for the Area Identifiers, thereby forming the areas, or alternatively an automatic formation process described here can be used by the connection owners. Non connection owners on a particular connection should simply follow the connection owner’s lead.
If the connection owner does not have an area on a particular connection yet, it may use an existing area from some other connection if and only if following suitability criteria are met:

- The current set of links covered by that area (calculated by traversing through the neighbor graph) is not more than TBD.
- The number of nodes in that area is not more than TBD.
- The area owner does not publish an Area Full TLV.

If nothing suitable is present, areas connected directly to other nodes within the area can be also considered. For them, the suitability criteria are:

- A node within current area exists which publishes Remote Area TLV with the Area Identifier of the area.
- No published Area Full TLV for the area.

If choosing to use a particular area, the node MUST wait random \([TBD1, TBD2]\) seconds before making the actual assignment, and ensure that the suitability criteria are still matched when it makes the assignment. If not, this process should be repeated again, starting from evaluating the candidates.

If no area is found at all, a new area should be created, with a random delay of \([TBD1, TBD2]\) seconds before announcing. At the end of the interval, the presence of available areas to join should be checked before publishing the Area Identifier TLV.

Once the area owner notices that the directly connected suitability criteria enumerated above are no longer filled by the local area (=it is too large), the area owner MUST publish an Area Full TLV. It MAY be removed at later point, but if and only if the area is substantially below the maximum desired size in terms of number of links and number of nodes.

If the owner of an area detects the presence of a Remote Area TLV with an Area Identifier identical to that of the area it is advertising and with an owner having a higher Node Identifier than itself, then the area owner MUST choose a new (random) Area Identifier.

TBD: Area Full TLV - no content, but net-wide.
7.4. Import/Export

There is no explicit exporting of TLVs; any TLV type that has highest bit set (0x8000) will be considered area-originated, and spread net-wide, as opposed to the default area-wide node-originated. It is important to note that currently node identifier of the originating node is lost as it transitions to another area (TBD), but within the area the originator is still visible.

Given the node is on an area boundary, for all areas it is in, it must recursively traverse all Remote Area TLVs announced within the area, and keep track of the shortest recursion depth at which a particular area is first encountered. The Node Identifier of the Remote Area TLV originator is used for tie-breaking, with the higher one preferred. If encountering Remote Area TLV with the local area’s Area Identifier, that TLV MUST NOT be recursed into to avoid loops.

For any areas for which the node is identified as the importer (by having shortest path of areas, or winning tie-break), the node MUST import Remote Area Content TLV from the first-hop remote area verbatim if there are other areas on the path. If the node is directly connected to the remote area, it MUST create and maintain Remote Area Content TLV which contains all TLVs marked for export.

When Remote Area Content TLV changes, or is no longer present in the "upstream" area, it must be also updated/removed by the importer.

TBD: Remote Area Content TLV - area id (+container for any exported TLVs from that area)

8. Security Considerations

TBD

9. IANA Considerations

TBD - TLVs values here + ADNCP-UDP-PORT, ADNCP-TCP-PORT

All-ADNCP-Nodes-4, All-ADNCP-Nodes-6

10. References

10.1. Normative references

[I-D.ietf-homenet-dncp]
10.2. Informative references

[I-D.ietf-homenet-prefix-assignment]

[I-D.irtf-nmrg-autonomic-network-definitions]

Appendix A. Open Issues

Should hierarchical PA be defined here or not? [I-D.ietf-homenet-prefix-assignment], with cross-area hierarchical extension, would facilitate even very large scale PA (with potentially multiple upstreams). Perhaps the current mention is enough.

Should areas importers / area ID choice TLVs include precedence value?

Should we include node-data signatures or not? They improve security, but are not visible across areas in any case - it would need per-TLV signature(!) in that case with a hefty footprint due to needing to include way to identify the public key too. So I think not.

Should some way to publish certificate id / raw public key be defined? So it can be verified that e.g. node identifier is really generated based on one. Perhaps..

Should some sort of more granular delta transfer scheme be defined? For a large network, the current scheme’s TLV set published by a single node can grow to substantial size. This may occur either here or in DNCP.
Appendix B. Changelog

draft-stenberg-anima-adncp-00: Initial version.

Appendix C. Draft Source

As usual, this draft is available at https://github.com/fingon/ietf-drafts/ in source format (with nice Makefile too). Feel free to send comments and/or pull requests if and when you have changes to it!

Appendix D. Acknowledgements

Thanks to Pierre Pfister, Mark Baugher and Steven Barth for their contributions to the draft.

Author’s Address

Markus Stenberg
Helsinki 00930
Finland

Email: markus.stenberg@iki.fi