ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2015

M. Behringer
S. Bjarnason
Balaji. BL
T. Eckert
Cisco

March 6, 2015

An Autonomic Control Plane
draft-behringer-anima-autonomic-control-plane-02

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

Copyright Notice

Table of Contents

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:

- o An operator can use it to log into remote devices, even if the data plane is misconfigured or unconfigured.

- o 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]

- o 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.

```
           autonomic node 1                  autonomic node 2
         ...................               ...................
   secure .                 .   secure      .                 .  secure
   tunnel :  +-----------+  :   tunnel      :  +-----------+  :  tunnel
   ..--------| ACP VRF   |---------------------| ACP VRF   |---------..
         : / \         / \   <--routing-->   / \         / \
         : \ /         \ /                   \ /         \ /
   ..--------| loopback  |---------------------| loopback  |---------..
         :  +-----------+  :               :  +-----------+  :
         :                 :               :                 :
         :   data plane    :...............:   data plane    :
         :                 :    link       :                 :
         :.................:               :.................:
```

Figure 1

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.

- o Globally valid Unique device identifier (UDI).
- o Link Local IPv6 address with its scope.
- o Trust information: The certificate chain, if available.
- o 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:

- o 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.
- o 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 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 unintential 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:

- o Prefix FC01::/8

- o 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.

- o 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:

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

- o When any changes happen in the topology, the routing protocol used in the ACP will automatically adapt to the changes and will continue to provide reachability to all devices.

- o 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:

- o The usage of domain certificates depends on a valid supporting PKI infrastructure. If the chain of trust of this PKI infrastructure is compromised, the security of the ACP is also compromised. This is typically under the control of the network administrator.

- o 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

- o Clarified that the ACP should be based on, and support only IPv6.
- o Clarified in intro that ACP is for both, between devices, as well as for access from a central entity, such as an NMS.
- o Added a section on how to connect an NMS system.
- o Clarified the hop-by-hop crypto nature of the ACP.
- o Added several references to GDNP as a candidate protocol.
- o 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]
Behringer, M., Carpenter, B., and T. Eckert, "A Reference Model for Autonomic Networking", draft-behringer-anima-reference-model-00 (work in progress), October 2014.

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

[I-D.carpenter-anima-gdn-protocol]
Carpenter, B. and B. Liu, "A Generic Discovery and Negotiation Protocol for Autonomic Networking", draft-carpenter-anima-gdn-protocol-02 (work in progress), February 2015.

[I-D.eckert-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Autonomic Network Stable Connectivity", draft-eckert-anima-stable-connectivity-00 (work in progress), October 2014.

[I-D.irtf-nmrg-an-gap-analysis]
Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-04 (work in progress), March 2015.

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

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 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

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2016

M. Behringer, Ed.
Cisco Systems
S. Bjarnason
Balaji. BL
T. Eckert
Cisco

June 30, 2015

An Autonomic Control Plane
draft-behringer-anima-autonomic-control-plane-03

Abstract

Autonomic functions need a control plane to communicate, which depends on some addressing and routing. This Autonomic Control Plane should ideally be self-managing, and as independent as possible of configuration. One application is a "virtual out of band channel" for communications over a network that is not configured or mis-configured. This document describes requirements and implementation options for an "Autonomic Control Plane".

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

Autonomic Networking in the IETF is currently being defined in the document [I-D.behringer-anima-reference-model]

Autonomic functions need a stable and robust infrastructure to communicate on. This infrastructure should be as robust as possible, and it should be re-usable by all autonomic functions. [RFC7575] calls it the "Autonomic Control Plane". This document defines the requirements and implementation options of an Autonomic Control Plane.

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 options for a self-forming, self-managing and self-protecting "Autonomic Control Plane" (ACP) which is inband on the network, yet as independent as possible of configuration, addressing and routing problems (for details how this achieved, see Section 5). 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:

- o Autonomic functions communicate over the ACP.

- o An operator can use it to log into remote devices, even if the data plane is misconfigured or unconfigured.

- o 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]

- o 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.

This document describes some use cases for the ACP in Section 2, it defines the requirements in Section 3, Section 4 gives an overview how an Autonomic Control Plane is constructed, and in Section 5 the detailed process is explained. 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.

2. Use Cases for an Autonomic Control Plane

2.1. An Infrastructure for Autonomic Functions

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

2.2. 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.

2.3. 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.

- 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. Requirements

The Autonomic Control Plane has the following requirements:

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

2. The ACP MUST have a separate address space from the data plane. Reason: traceability, debug-ability, separation from data plane, security (can block at edge)

3. The ACP MUST use autonomically managed address space. Reason: easy bootstrap and setup ("autonomic"); robustness (admin can't mess things up so easily). ULA seems like a good choice for 1 and 2.

4. The ACP MUST be generic. Usable by all the functions and protocols of the AN infrastructure. MUST NOT be tied to a particular protocol.

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

The default mode of operation of the ACP is hop-by-hop, because this interaction can be built on IPv6 link local addressing, which is autonomic, and has no dependency on configuration (requirement 1). It may be necessary to have end-to-end connectivity in some cases, for example to provide an end-to-end security association for some protocols. This is possible, but then has a dependency on routable address space.

4. Overview

The Autonomic Control Plane is constructed in the following way (for details, see Section 5):

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

- 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.

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

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

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

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

The following figure illustrates the ACP.

```
           autonomic node 1                  autonomic node 2
         ...................               ...................
   secure .                 .   secure      .                 .  secure
   tunnel :  +-----------+  :   tunnel      :  +-----------+  :  tunnel
   ..--------| ACP VRF   |---------------------| ACP VRF   |---------..
         : / \         / \   <--routing-->   / \         / \ :
         : \ /         \ /                   \ /         \ / :
   ..--------| loopback  |---------------------| loopback  |---------..
         :  +-----------+  :               :  +-----------+  :
         :                 :               :                 :
         :   data plane    :...............:   data plane    :
         :                 :    link       :                 :
         :.................:               :.................:
```

Figure 1

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.

An alternative ACP design can be achieved without the VRFs. In this case, the autonomic virtual addresses are part of the data plane, and subject to routing, filtering, QoS, etc on the data plane. The secure tunnels are in this case used by traffic to and from the autonomic address space. They are still required to provide the authentication function for all autonomic packets.

5. 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.

5.1. 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].)

5.2. Adjacency Discovery

Adjacency discovery exchanges identity information about neighbors, either the UDI or, if present, the domain certificate (see Section 5.1. 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.

- o Globally valid Unique device identifier (UDI).
- o Link Local IPv6 address with its scope.
- o Trust information: The certificate chain, if available.
- o 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.

5.3. 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:

- o 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.
- o 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.

5.4. 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 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.

5.5. 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:

- o 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.

- o 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 5.8.

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.

5.6. 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 unintential 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.

5.7. 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 5.6.

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

- Prefix FC01::/8

- o 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 6 for a discussion on merging domains.

- o 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.

5.8. 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.

5.9. 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.

6. Self-Healing Properties

The ACP is self-healing:

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

- o When any changes happen in the topology, the routing protocol used in the ACP will automatically adapt to the changes and will continue to provide reachability to all devices.

- o 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 5.7).

7. Self-Protection Properties

As explained in Section 5, 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.

8. 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 5.9 for more details on how to connect an NMS host into the ACP.

9. 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:

- o The usage of domain certificates depends on a valid supporting PKI infrastructure. If the chain of trust of this PKI infrastructure is compromised, the security of the ACP is also compromised. This is typically under the control of the network administrator.

- o 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.

10. IANA Considerations

This document requests no action by IANA.

11. 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.

12. Change log [RFC Editor: Please remove]

12.1. Initial version

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

12.2. version 00

Initial version of the anima document; only minor edits.

12.3. version 01

- o Clarified that the ACP should be based on, and support only IPv6.
- o Clarified in intro that ACP is for both, between devices, as well as for access from a central entity, such as an NMS.
- o Added a section on how to connect an NMS system.
- o Clarified the hop-by-hop crypto nature of the ACP.
- o Added several references to GDNP as a candidate protocol.

- o Added a discussion on network split and merge. Although, this should probably go into the certificate management story longer term.

12.4. 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

12.5. version 03

- o Took out requirement for IPv6 --> that’s in the reference doc.
- o Added requirement section.
- o Changed focus: more focus on autonomic functions, not only virtual out of band. This goes a bit throughout the document, starting with a changed abstract and intro.

13. References

[I-D.behringer-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Liu, B., Jeff, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-behringer-anima-reference-model-03 (work in progress), June 2015.

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

[I-D.carpenter-anima-gdn-protocol]
Carpenter, B. and B. Liu, "A Generic Discovery and Negotiation Protocol for Autonomic Networking", draft-carpenter-anima-gdn-protocol-04 (work in progress), June 2015.

[I-D.eckert-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-eckert-anima-stable-connectivity-01 (work in progress), March 2015.

[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.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, June 2015.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, June 2015.

Authors' Addresses

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

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

ANIMA
Internet-Draft
Intended status: Informational
Expires: April 20, 2015

M. Behringer, Ed.
Cisco
B. Carpenter
Univ. of Auckland
T. Eckert
Cisco

October 17, 2014

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

Copyright Notice

Table of Contents

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:

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

- o 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.

- o 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.

```
+-------------------------------------------------------------+
|                                                             |
| +-----------+        +------------+        +------------+  |
| | Autonomic |        | Autonomic  |        | Autonomic  |  |
| | Service   |        | Service    |        | Service    |  |
| | Agent 1   |        | Agent 2    |        | Agent 3    |  |
| +-----------+        +------------+        +------------+  |
|       ^                    ^                    ^         |
|       |                    |                    |         |
|       V                    V                    V         |
|-------------------------------------------------------------|
| Autonomic Networking Infrastructure                         |
|    - Data structures (ex: certificates, peer information)  |
|    - discovery, negotiation and synchronisation functions  |
|    - intent distribution                                    |
|    - aggregated reporting and feedback loops                |
|    - routing                                                |
|-------------------------------------------------------------|
|           Standard Operating System Functions               |
+-------------------------------------------------------------+
```

Figure 1

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 neigbhour 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 by 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]
Behringer, M., Bjarnason, S., BL, B., and T. Eckert, "An Autonomic Control Plane", draft-behringer-autonomic-control-plane-00 (work in progress), June 2014.

[I-D.carpenter-anima-gdn-protocol]
Carpenter, B., Jiang, S., and B. Liu, "A Generic Discovery and Negotiation Protocol for Autonomic Networking", draft-carpenter-anima-gdn-protocol-00 (work in progress), October 2014.

[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-04 (work in progress), October 2014.

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

Authors' Addresses

Michael H. Behringer (editor)
Cisco

Email: mbehring@cisco.com

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

Email: brian.e.carpenter@gmail.com

Toerless Eckert
Cisco

Email: eckert@cisco.com

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

M. Behringer, Ed.
Cisco Systems
B. Carpenter
Univ. of Auckland
T. Eckert
Cisco
L. Ciavaglia
Alcatel Lucent
B. Liu
Huawei Technologies
J. Nobre
Federal University of Rio Grande do Sul
J. Strassner
Huawei Technologies
October 16, 2015

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

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

2. The Network View

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

Figure 1 shows the high level view of an Autonomic Network. It consists of a number of autonomic nodes, which interact directly with

each other. Those autonomic nodes provide a common set of capabilities across the network, called the "Autonomic Networking Infrastructure" (ANI). The ANI provides functions like naming, addressing, negotiation, synchronization, discovery and messaging.

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

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

Figure 1: High level view of an Autonomic Network

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

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

3. The Autonomic Network Element

3.1. Architecture

This section describes an autonomic network element and its internal architecture. The reference model explained in the document "Autonomic Networking - Definitions and Design Goals" [RFC7575] shows the sources of information that an autonomic service agent can leverage: Self-knowledge, network knowledge (through discovery), Intent, and feedback loops. Fundamentally, there are two levels

inside an autonomic node: the level of Autonomic Service Agents, and the level of the Autonomic Networking Infrastructure, with the former using the services of the latter. Figure 2 illustrates this concept.

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

Figure 2: Model of an autonomic node

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

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

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

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

4. The Autonomic Networking Infrastructure

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

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

4.1. Naming

4.1.1. Naming requirements

- o Representing each device

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

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

- o Uniqueness

  The names MUST NOT collide within one autonomic domain.

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

- o Semantic Encoding

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

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

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

- o Consistency

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

4.1.2. Proposed Mechanisms

——

- o Structured Naming Pattern

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

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

- o Advertised Common Fields

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

- o Self-generated Fields

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

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

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

4.2. Addressing

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

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

An autonomic addressing scheme has the following requirements:

- Zero-touch for simple networks: Simple networks should have complete self-management of addressing, and not require any central address management, tools, or address planning.

- Low-touch for complex networks: If complex networks require operator input for autonomic address management, it should be limited to high level guidance only, expressed in Intent.

- Flexibility: The addressing scheme must be flexible enough for nodes to be able to move around, for the network to grow, split and merge.

- Robustness: It should be as hard as possible for an administrator to negatively affect addressing (and thus connectivity) in the autonomic context.

- Support for virtualization: Autonomic Nodes may support Autonomic Service Agents in different virtual machines or containers. The addressing scheme should support this architecture.

- Simplicity: To make engineering simpler, and to give the human administrator an easy way to trouble-shoot autonomic functions.

- Scale: The proposed scheme should work in any network of any size.

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

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

4.3. Discovery

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

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

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

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

4.4. Signaling Between Autonomic Nodes

Autonomic nodes must communicate with each other, for example to negotiate and/or synchronize technical objectives (i.e., network parameters) of any kind and complexity. This requires some form of signaling between autonomic nodes. Autonomic nodes implementing a

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

4.5. Intent Distribution

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

4.6. Routing

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

4.7. The Autonomic Control Plane

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

5. Functional Overview

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

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

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

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

The adjacency table is fed by the following inputs:

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

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

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

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

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

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

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

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

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

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

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

6. Security and Trust Infrastructure

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

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

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

6.1. Public Key Infrastructure

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

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

6.2. Domain Certificate

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

6.3. The MASA

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

6.4. Sub-Domains (*)

Explain how sub-domains are handled. (tbc)

6.5. Cross-Domain Functionality (*)

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

7. Autonomic Service Agents (ASA)

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

7.1. General Description of an ASA

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

7.2. Specific ASAs for the Enrolment Process

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

7.2.1. The Enrolment ASA

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

7.2.2. The Enrolment Proxy ASA

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

7.2.3. The Registrar ASA

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

8. Management and Programmability

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

8.1. How an AN Network Is Managed

Autonomic management usually co-exists with traditional management methods in most networks. Thus, autonomic behavior will be defined for individual functions in most environments. In fact, the co-existence is twofold: autonomic functions can use traditional methods

and protocols (e.g., SNMP and NETCONF) to perform management tasks; and autonomic functions can conflict with behavior enforced by the same traditional methods and protocols.

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

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

8.2. Intent (*)

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

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

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

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

Some examples of parameters for intents are:

- o Model version: The version of the model used to define the intent.
- o Domain: The network scope in which the intent has effect.
- o Name: The name of the intent which describes the intent for human operators.
- o Version: The version of the intent, which is primarly used to control intent updates.
- o Signature: The signature is used as a security mechanism to provide authentication, integrity, and non-repudiation.
- o Timestamp: The timestamp of the creation of the intent using the format supported by the IETF [TBC].
- o Lifetime: The lifetime in which the intent may be observed. A special case of the lifetime is the definition of permanent intents.

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

8.3. Aggregated Reporting (*)

Autonomic Network should minimize the need for human intervention. In terms of how the network should behave, this is done through an autonomic intent provided by the human administrator. In an analogous manner, the reports which describe the operational status of the network should aggregate the information produced in different network elements in order to present the effectiveness of autonomic

intent enforcement. Therefore, reporting in an autonomic network should happen on a network-wide basis [RFC7575]. The information gathering and the reporting delivery should be done through the autonomic control plane.

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

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

8.4. Feedback Loops to NOC(*)

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

8.5. Control Loops (*)

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

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

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

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

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

8.6. APIs (*)

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

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

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

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

8.7. Data Model (*)

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

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

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

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

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

9. Coordination Between Autonomic Functions (*)

9.1. The Coordination Problem (*)

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

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

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

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

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

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

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

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

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

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

9.2. A Coordination Functional Block (*)

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

A common coordination function requires:

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

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

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

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

10. Security Considerations

10.1. Threat Analysis

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

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

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

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

on any unprotected Layer 2 link, and to insider attacks even on protected Layer 2 links.

11. IANA Considerations

This document requests no action by IANA.

12. Acknowledgements

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

13. References

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

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

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

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

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

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

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

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

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <http://www.rfc-editor.org/info/rfc7575>.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, June 2015, <http://www.rfc-editor.org/info/rfc7576>.

Authors' Addresses

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

Email: mbehring@cisco.com

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

Email: brian.e.carpenter@gmail.com

Toerless Eckert
Cisco

Email: eckert@cisco.com

Laurent Ciavaglia
Alcatel Lucent
Route de Villejust
Nozay 91620
France

Email: laurent.ciavaglia@alcatel-lucent.com

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

Email: leo.liubing@huawei.com

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

Email: jcnobre@inf.ufrgs.br

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

Email: john.sc.strassner@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 23, 2015

B. Carpenter
Univ. of Auckland
B. Liu
Huawei Technologies Co., Ltd

February 19, 2015

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

Copyright Notice

Table of Contents

1. Introduction

The success of the Internet has made IP-based networks bigger and more complicated. Large-scale ISP and enterprise networks have become more and more problematic for human based management. Also, operational costs are growing quickly. Consequently, there are increased requirements for autonomic behavior in the networks. General aspects of autonomic networks are discussed in [I-D.irtf-nmrg-autonomic-network-definitions] and [I-D.irtf-nmrg-an-gap-analysis]. In order to fulfil 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:

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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.

- o 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 neigbor 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.

- o 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.

- o 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:

- o the identity of a GDNP message sender can be verified by a recipient.
- o the integrity of a GDNP message can be checked by the recipient of the message.
- o 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:

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

- o 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:

- o 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.

- o 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) x (1 - 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

- o 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

- o GDNP Listen Port (TBD3)

  A UDP and TCP port that every GDNP-enabled network device always listens to.

- o GDNP_DEF_TIMEOUT (60000 milliseconds)

  The default timeout used to determine that a discovery or negotiation has failed to complete.

- o 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 Certifcate 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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| MESSAGE_TYPE  |               Session ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                   Device Certificate Tag                      |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Options  (variable length)             |
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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:

- o a discovery objective option (Section 3.9.1).

- o a negotiation objective option (Section 3.9.1) to indicate to the discovery target that it MAY directly reply to the discovery initiatior with a NEGOTIATION message for rapid processing, if it could act as the corresponding negotiation counterpart. The

sender of such a DISCOVERY message MUST initialize a negotiation timer and loop count in the same way as a REQUEST message (Section 3.7.4).

- o 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
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         option-data                           |
|                       (option-len octets)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_DIVERT         |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Locator Option(s) of Diversion Device(s)          |
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

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

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.

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

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.

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

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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION Certificate      |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                     Certificate (variable length)             .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    OPTION_LOCATOR_IPV4ADDR    |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          IPv4-Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   OPTION_LOCATOR_IPV6ADDR     |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                          IPv6-Address                         |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_FQDN           |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Fully Qualified Domain Name                 |
|                       (variable length)                       |
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_XXX            |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   loop-count  |    flags      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          value                |
.                                   (variable length)           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_vendor         |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             PEN                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   loop-count  |    flags      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          value                |
.                                   (variable length)           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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):

- o 1. UDP vs TCP: For now, this specification suggests UDP and TCP as message transport mechanisms. This is not clarified yet. UDP is good for short conversations, is necessary for multicast discovery, and generally fits the discovery and divert scenarios well. However, it will cause problems with large messages. TCP is good for stable and long sessions, with a little bit of time consumption during the session establishment stage. If messages exceed a reasonable MTU, a TCP mode will be required in any case. This question may be affected by the security discussion.

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

- o 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.

- o 3. DoS Attack Protection needs work.

- o 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.

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

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

- o 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 succesfully 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:

Type	Name	RFCs
0	Reserved	this document
1	Discovery	this document
2	Response	this document
3	Request Message	this document
4	Negotiation Message	this document
5	Negotiation-end Message	this document
6	Confirm-waiting Message	this document

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:

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

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:

Name	Value	RFCs
Reserved	0x0000	this document
SHA-1	0x0001	this document
SHA-256	0x0002	this document

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:

Name	Value	RFCs
Reserved	0x0000	this document
RSASSA-PKCS1-v1_5	0x0001	this document

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 Requirement Levels", BCP 14, RFC 2119, March 1997.

[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 Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

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]
Behringer, M., Chaparadza, R., Petre, R., Li, X., and H. Mahkonen, "IP based Generic Control Protocol (IGCP)", draft-chaparadza-intarea-igcp-00 (work in progress), July 2011.

[I-D.eckert-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Autonomic Network Stable Connectivity", draft-eckert-anima-stable-connectivity-00 (work in progress), October 2014.

[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]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", draft-ietf-homenet-dncp-00 (work in progress), January 2015.

[I-D.ietf-homenet-hncp]
Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", draft-ietf-homenet-hncp-03 (work in progress), January 2015.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-04 (work in progress), January 2015.

[I-D.irtf-nmrg-an-gap-analysis]
Jiang, S., Carpenter, B., and M. Behringer, "Gap Analysis for Autonomic Networking", draft-irtf-nmrg-an-gap-analysis-03 (work in progress), December 2014.

[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.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.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2608] Guttman, E., Perkins, C., Veizades, J., and M. Day, "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 Tunnels", RFC 3209, December 2001.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

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

[RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, November 2005.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

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

Home Networking Control Protocol (HNCP) [I-D.ietf-homenet-hncp] for configuring Homenet routers.

Specific features of DNCP include:

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2015

B. Carpenter
Univ. of Auckland
B. Liu
Huawei Technologies Co., Ltd

June 20, 2015

A Generic Discovery and Negotiation Protocol for Autonomic Networking
draft-carpenter-anima-gdn-protocol-04

Abstract

This document establishes requirements for a signaling protocol that enables autonomic devices and autonomic service agents to dynamically discover peers, 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

Copyright Notice

Table of Contents

1. Introduction

The success of the Internet has made IP-based networks bigger and more complicated. Large-scale ISP and enterprise networks have become more and more problematic for human based management. Also, operational costs are growing quickly. Consequently, there are increased requirements for autonomic behavior in the networks. General aspects of autonomic networks are discussed in [RFC7575] and [RFC7576]. A reference model for autonomic networking is given in [I-D.behringer-anima-reference-model]. In order to fulfil autonomy, devices that embody autonomic service agents have specific signaling requirements. In particular they need to discover each other, to synchronize state with each other, and to negotiate parameters and resources directly with each other. There is no 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 non-autonomic network. The atomic unit of synchronization or negotiation is referred to as a technical objective, i.e, a configurable parameter or set of parameters (defined more precisely in Section 3.1).

Following this Introduction, Section 2 describes the requirements for 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 managed networks. However, when a device starts up with no pre-configuration, it has no knowledge of the topology. The protocol itself is capable of being used in a small and/or flat network structure such as a small office or home network as well as 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 must run in a secure and strongly authenticated environment.

It is understood that in realistic deployments, not all devices will support GDNP. It is expected that some autonomic service agents will directly 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. The primary user of the protocol is an autonomic service agent (ASA), so the requirements are mainly expressed as the features needed by an ASA. A single physical device might contain several ASAs, and a single ASA might manage several technical objectives.

2.1. Requirements for Discovery

1. ASAs may be designed to manage anything, as required in Section 2.2. A basic requirement is therefore that the protocol can represent and discover any kind of technical objective among arbitrary subsets of participating nodes.

In an autonomic network we must assume that when a device starts up it has no information about any peer devices, the network structure, or what specific role it must play. The ASA(s) inside the device are in the same situation. In some cases, when a new application session starts up within a device, the device or ASA may again lack information about relevant peers. It might be necessary to set up resources on multiple other devices, coordinated and matched to each other so that there is no wasted resource. Security settings might also need updating to allow for the new device or user. The relevant peers may be different for different technical objectives. Therefore discovery needs to be repeated as often as necessary to find peers capable of acting as counterparts for each objective that a discovery

initiator needs to handle. From this background we derive the next three requirements:

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

3. When an ASA starts up, it must require no information about any peers in order to discover them.

4. If an ASA supports multiple technical objectives, relevant peers may be different for different discovery objectives, so discovery needs to be repeated to find counterparts for each objective. Thus, there must be a mechanism by which an ASA can separately discover peer ASAs for each of the technical objectives that it needs to manage, whenever necessary.

5. Following discovery, an ASA will normally perform negotiation or synchronization for the corresponding objectives. The design should allow for this by associating discovery, negotiation and synchronization objectives. It may provide an optional mechanism to combine discovery and negotiation/synchronization in a single call.

6. Some objectives may only be significant on the local link, but others may be significant across the routed network and require off-link operations. Thus, the relevant peers might be immediate neighbors on the same layer 2 link, or they might be more distant and only accessible via layer 3. The mechanism must therefore provide both on-link and off-link discovery of ASAs supporting specific technical objectives.

7. The discovery process should be flexible enough to allow for special cases, such as the following:

- In some networks, as mentioned above, there will be some hierarchical structure, at least for certain synchronization or negotiation objectives, but this is unknown in advance. The discovery protocol must therefore operate regardless of hierarchical structure, which is an attribute of individual technical objectives and not of the autonomic network as a whole. This is part of the more general requirement to discover off-link peers.

- During initialisation, a device must be able to establish mutual trust with the rest of the network and join an authentication mechanism. Although this will inevitably start with a discovery action, it is a special case precisely because trust is not yet

established. This topic is the subject of [I-D.pritikin-anima-bootstrapping-keyinfra]. We require that once trust has been established for a device, all ASAs within the device inherit the device's credentials and are also trusted.

- o Depending on the type of network involved, discovery of other central functions might be needed, such as a source of Intent distribution [RFC7575] or the Network Operations Center (NOC) [I-D.eckert-anima-stable-connectivity]. The protocol must be capable of supporting such discovery during initialisation, as well as discovery during ongoing operation.

8. The discovery process must not generate excessive (multicast) traffic and must take account of sleeping nodes in the case of a resource-constrained network [RFC7228].

2.2. Requirements for Synchronization and Negotiation Capability

As background, consider the example of routing protocols, the closest approximation to autonomic networking already in widespread use. Routing protocols use a largely autonomic model based on distributed devices that communicate repeatedly with each other. The focus is reachability, so current routing protocols mainly consider simple link status, i.e., up or down, and an underlying assumption is that all nodes need a consistent view of the network topology in order for the routing algorithm to converge. Thus, routing is mainly based on information synchronization between peers, rather than on bi-directional negotiation. Other information, such as latency, congestion, capacity, and particularly unused capacity, would be helpful to get better path selection and utilization rate, but is not normally used in distributed routing algorithms. Additionally, autonomic networks need to be able to manage many more dimensions, such as security settings, power saving, load balancing, etc. 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. In general, these parameters do not apply to all participating nodes, but only to a subset.

9. A basic requirement for the protocol is therefore the ability to represent, discover, synchronize and negotiate almost any kind of network parameter among arbitrary subsets of participating nodes.

10. Negotiation is a request/response process that 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 these must be defined specifically for each use case, the protocol should have some general mechanisms in support of loop and deadlock prevention, such as hop count limits or timeouts.

11. Synchronization might concern small groups of nodes or very large groups. Different solutions might be needed at different scales.

12. To avoid "reinventing the wheel", the protocol should be able to carry the message formats used by existing configuration protocols (such as NETCONF/YANG) in cases where that is convenient.

13. 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 therefore follows that the protocol, as part of the Autonomic Networking Infrastructure, must be capable of running in any device that would otherwise need human intervention.

14. Human intervention in large networks is often replaced by use of a top-down network management system (NMS). It therefore follows that the protocol, as part of the Autonomic Networking Infrastructure, must be capable of running in any device that would otherwise be managed by an NMS, and that it can co-exist with an NMS, and with protocols such as SNMP and NETCONF.

15. Some features are expected to be implemented by individual ASAs, but the protocol must be general enough to allow them:

- o Dependencies and conflicts: 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 and conflicts among negotiation or synchronization procedures. Resolving dependencies and conflicts is a matter for the individual ASAs involved. To allow this, there need to be clear boundaries and convergence mechanisms for negotiations. Also some mechanisms are needed to avoid loop dependencies. In such a case, the protocol’s role is limited to signaling between ASAs.

- o Recovery from faults and identification of faulty devices should be as automatic as possible. The protocol’s role is limited to the ability to handle discovery, synchronization and negotiation

at any time, in case an ASA detects an anomaly such as a negotiation counterpart failing.

- o 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.

- o Management logging, monitoring, alerts and tools for intervention are required. However, these can only be features of individual ASAs. 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].

16. The protocol will be able to deal with a wide variety of technical objectives, covering any type of network parameter. Therefore the protocol will need 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.

2.3. Specific Technical Requirements

17. It should be convenient for ASA designers to define new technical objectives and for programmers to express them, without excessive impact on run-time efficiency and footprint. The classes of device in which the protocol might run is discussed in [I-D.behringer-anima-reference-model].

18. The protocol should be extensible in case the initially defined discovery, synchronization and negotiation mechanisms prove to be insufficient.

19. 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.

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

21. It must also be possible for an external discovery mechanism to be used, if appropriate for a given technical objective. In other

words, GDNP discovery must not be a prerequisite for GDNP negotiation or synchronization; the prerequisite is discovering a peer’s locator by any method.

22. ASAs and the signaling protocol engine need to run asynchronously when wait states occur.

23. Intent: There must be provision for general Intent rules to be applied by all devices in the network (e.g., security rules, prefix length, resource sharing rules). However, Intent distribution might not use the signaling protocol itself, but its design should not exclude such use.

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

25. The protocol needs to be fully secured against forged messages and man-in-the middle attacks, and secured as much as reasonably possible against denial of service attacks. It needs to be capable of encryption in order to resist unwanted monitoring, although this capability may not be required in all deployments. However, it is not required that the protocol itself provides these security features; it may depend on an existing secure environment.

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.

This document uses terminology defined in [RFC7575].

The following additional terms are used throughout this document:

- Discovery: a process by which an ASA discovers peers according to a specific discovery objective. The discovery results may be different according to the different discovery objectives. The

discovered peers may later be used as negotiation counterparts or as sources of synchronization data.

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

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

- o Technical Objective (usually abbreviated as Objective): A technical objective is a configurable parameter or set of parameters 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 ASA 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 contain an ASA 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 ASAs.

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

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

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

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

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

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

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

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 early version was 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. This whole specification is subject to change as a result.

- o 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 technical objectives and the different pairs of counterparts.

- o The protocol is expected to form part of an Autonomic Networking Infrastructure [I-D.behringer-anima-reference-model]. It will provide services to ASAs via a suitable application programming interface, which will reflect the protocol elements but will not necessarily be in one-to-one correspondence to them. It is expected that the protocol engine and each ASA will run as independent asynchronous processes.

- o 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 assumed to run within an existing secure environment with strong authentication.

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

- o 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.

  * GDNP discovery is appropriate for efficient discovery of GDNP peers and allows a rapid mode of operation described in Section 3.3.3. For some parameters, especially those concerned with application layer services, a text-based discovery mechanism such as DNS Service Discovery [I-D.ietf-dnssd-requirements] or Service Location Protocol [RFC2608] might be more appropriate. The choice is left to the designers of individual ASAs.

- o 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.

- o A flexible model for synchronization

  GDNP supports bilateral synchronization, which could be used to perform synchronization among a small number of nodes. It also supports an unsolicited flooding mode when large groups of nodes, possibly including all autonomic nodes, need data for the same technical objective.

* There may be some network parameters for which a more traditional flooding mechanism such as ADNCP [I-D.ietf-homenet-dncp] [I-D.stenberg-anima-adncp] is considered more appropriate. GDNP can coexist with ADNCP.

o 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 multi-party negotiations by indirect steps.

o 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.

o Self-aware network device

Every autonomic device will be pre-loaded with various functions and ASAs and will be aware of its own capabilities, typically decided by the hardware, firmware or pre-installed software. Its exact role may depend on Intent and 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.

o Requests and responses in negotiation procedures

The initiator can negotiate with its relevant negotiation counterpart ASAs, 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 ASAs.

- o 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 ASA for each negotiation objective. It may be an implementation choice, a pre-configurable parameter, or network Intent. These choices might vary between different types of ASA. 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 network Intent.

3.3. GDNP Protocol Basic Properties and Mechanisms

3.3.1. Required External Security Mechanism

The protocol SHOULD run within a secure Autonomic Control Plane (ACP) [I-D.behringer-anima-autonomic-control-plane]. The procedure for establishing the ACP MUST provide a flag indicating to GDNP that the ACP has been established.

If there is no ACP, the protocol MUST use TLS [RFC5246] or DTLS [RFC6347] for all messages, based on a local Public Key Infrastructure (PKI) [RFC5280] managed within the autonomic network itself.

Link-local multicast is used for discovery messages. These cannot be secured, but responses to discovery messages MUST be secured. However, during initialisation, before a node has joined the applicable trust infrastructure, e.g., [I-D.pritikin-anima-bootstrapping-keyinfra], it might be impossible to secure certain messages. Such messages MUST be limited to the strictly necessary minimum.

3.3.2. Transport Layer Usage

The protocol is capable of running over UDP or TCP, except for link-local multicast discovery messages, which can only run over UDP and MUST NOT be fragmented, and therefore cannot exceed the link MTU size.

When running within a secure ACP, UDP SHOULD be used for messages not exceeding the minimum IPv6 path MTU, and TCP MUST be used for longer messages. In other words, IPv6 fragmentation is avoided. If a node receives a UDP message but the reply is too long, it MUST open a TCP connection to the peer for the reply.

When running without an ACP, TLS MUST be supported and used by default, except for multicast discovery messages. DTLS MAY be supported as an alternative but the details are out of scope for this document.

For all transport protocols, the GDNP protocol listens to the GDNP Listen Port (Section 3.4).

3.3.3. 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 3.6.2) or request (Section 3.6.4) message, the recipient ASA should return a message in which it either indicates itself as a discovery responder or diverts the initiator towards another more suitable ASA.

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 ASA the initiator will negotiate with.

- o Discovery Procedures

Discovery starts as an on-link operation. The Divert option can tell the discovery initiator to contact an off-link ASA for that discovery objective. 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 an ASA in the neighbor device supports the requested discovery objective, it MAY respond with a Response message (Section 3.6.3) with locator option(s). Otherwise, if the neighbor has cached information about an ASA that supports the requested discovery objective (usually because it discovered the same objective before), it SHOULD respond with a 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 3.5). An exponential backoff SHOULD be used for subsequent repetitions, in order to mitigate possible denial of service attacks.

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 that MAY be modified by network Intent.

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 MUST limit the total rate at which it relays discovery messages to a reasonable value, in order to mitigate possible denial of service attacks. 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. These precautions avoid discovery loops.

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

- o 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 gateway, which is in principle closer to the right counterpart. Finally the right counterpart responds to start the negotiation or synchronization process.

- o 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 3.3.4. A similar mechanism is defined for synchronization in Section 3.3.5.

3.3.4. Negotiation Procedures

A negotiation initiator sends a negotiation request to a counterpart ASA, 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 ASAs.

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 ASAs, 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 Intent.

3.3.5. Synchronization Procedure

A synchronization initiator sends a synchronization request to a counterpart, 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. For large groups of

nodes requiring the same data, synchronization flooding is available. For this, 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 is 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].

A GDNP device with multiple link-layer interfaces (typically a router) MUST support synchronization flooding on all interfaces. If it receives a multicast unsolicited Response message on a given interface, it MUST relay it by re-issuing the same Response message on its other interfaces. However, it MUST limit the total rate at which it relays Response messages to a reasonable value, in order to mitigate possible denial of service attacks. It MUST cache the Session ID value of each relayed discovery message and, to prevent loops, MUST NOT relay a Response message which carries such a cached Session ID. These precautions avoid synchronization loops.

Note that this mechanism is unreliable in the case of sleeping nodes. Sleeping nodes that require an objective subject to synchronization flooding SHOULD periodically initiate normal synchronization for that objective.

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(s). 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 Intent.

3.4. GDNP Constants

- o ALL_GDNP_NEIGHBOR

  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

- o GDNP Listen Port (TBD3)

  A UDP and TCP port that every GDNP-enabled network device always listens to.

- o GDNP_DEF_TIMEOUT (60000 milliseconds)

  The default timeout used to determine that a discovery or negotiation has failed to complete.

- o GDNP_DEF_LOOPCT (6)

  The default loop count used to determine that a negotiation has failed to complete.

3.5. 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].

3.6. 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.

3.6.1. GDNP Message Format

GDNP messages share an identical fixed format header and a variable format area for options. GDNP message headers and options are in the type-length-value (TLV) format defined in DNCP (see Section "Type-Length-Value Objects" in [I-D.ietf-homenet-dncp]).

Every GDNP message carries a Session ID. Options are presented serially in the options field, with padding to 4-byte alignment.

The following diagram illustrates the format of GDNP messages:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          MESSAGE_TYPE         |               4               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Reserved   |               Session ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Options  (variable length)             |
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MESSAGE_TYPE: Identifies the GDNP message type. 16-bit.

Reserved: Set to zero, ignored on receipt. 8-bit.

Session ID: Identifies this GDNP session, as defined in Section 3.5. 24-bit.

Options: GDNP Options carried in this message. Options are defined starting at Section 3.7.

3.6.2. Discovery Message

DISCOVERY (MESSAGE_TYPE = G1):

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:

o a discovery objective option (Section 3.8.1).

- o a negotiation objective option (Section 3.8.1) to indicate to the discovery target that it MAY directly reply to the discovery initiatior with a NEGOTIATION message for rapid processing, if it could act as the corresponding negotiation counterpart. The sender of such a DISCOVERY message MUST initialize a negotiation timer and loop count in the same way as a REQUEST message (Section 3.6.4).

- o one or more synchronization objective options (Section 3.8.1) 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.

3.6.3. Response Message

RESPONSE (MESSAGE_TYPE = G2):

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 3.7.7) 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 3.7.2) 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, in the form of GDNP Option(s) for the specific synchronization objective(s).

3.6.4. Request Message

REQUEST (MESSAGE_TYPE = G3):

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 3.6.7), 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.

3.6.5. Negotiation Message

NEGOTIATION (MESSAGE_TYPE = G4):

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.

3.6.6. Negotiation-ending Message

NEGOTIATION-ENDING (MESSAGE_TYPE = G5):

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 3.7.3 and Section 3.7.4. It could be sent either by the requesting node or the responding node.

3.6.7. Confirm-waiting Message

CONFIRM-WAITING (MESSAGE_TYPE = G6):

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 3.7.5).

3.7. GDNP General Options

This section defines the GDNP general options for the negotiation and synchronization protocol signaling. Additional option types are reserved for GDNP general options defined in the future.

3.7.1. Format of GDNP Options

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         option-data                           |
|                       (option-len octets)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

3.7.2. 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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_DIVERT         |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Locator Option(s) of Diversion Target(s)          |
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option-code: OPTION_DIVERT (G32).

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 3.7.7) that point to diverted destination target(s).

3.7.3. 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.

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

Option-code: OPTION_ACCEPT (G33)

Option-len: 0

3.7.4. 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.

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

Option-code: OPTION_DECLINE (G34)

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.

3.7.5. 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 3.6.4).

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.

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

Option-code: OPTION_WAITING (G35)

Option-len: 4, in octets

Time: Time in milliseconds

3.7.6. Device Identity Option

The Device Identity option carries the identities of the sender and of the domain(s) that it belongs to. The format of the Device Identity 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 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION_DEVICE_ID        |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                     Identities (variable length)              .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option-code: OPTION_DEVICE_ID (G36)

Option-len: Length of identities in octets

Identities: A variable-length field containing the device identity and one or more domain identities. The format is not yet defined.

Note: Currently this option is a placeholder. It might be removed or modified.

3.7.7. Locator Options

These locator options are used to present reachability information for an ASA, a device or an interface. 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.

3.7.7.1. Locator IPv4 address option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    OPTION_LOCATOR_IPV4ADDR    |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          IPv4-Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option-code: OPTION_LOCATOR_IPV4ADDR (G37)

Option-len: 4, in octets

IPv4-Address: The IPv4 address locator of the target

Note: If an operator has internal network address translation for IPv4, this option MUST NOT be used within the Divert option.

3.7.7.2. Locator IPv6 address option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   OPTION_LOCATOR_IPV6ADDR     |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                          IPv6-Address                         |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option-code: OPTION_LOCATOR_IPV6ADDR (G38)

Option-len: 16, in octets

IPv6-Address: The IPv6 address locator of the target

Note: A link-local IPv6 address MUST NOT be used when this option is used within the Divert option.

3.7.7.3. Locator FQDN option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_FQDN           |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Fully Qualified Domain Name                |
|                        (variable length)                      |
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option-code: OPTION_FQDN (G39)

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

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

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.

3.8. Objective Options

3.8.1. 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:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_XXX            |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   flags       |   loop-count  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          value                |
.                                   (variable length)           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

option-len: The total length in octets.

flags: Flag bits.

Bit 0 (D bit): set if this objective is valid for GDNP discovery operations.

Bit 1 (N bit): set if this objective is valid for GDNP negotiation operations.

Bit 2 (S bit): set if this objective is valid for GDNP synchronization operations.

Bits 3~7: reserved, set to zero and ignored on reception.

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

3.8.2. General Considerations for Objective Options

Objective Options MUST be assigned an option type greater than G63 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.

3.8.3. 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.

3.8.4. Vendor Specific Objective Options

Option codes G128~159 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.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         OPTION_vendor         |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              PEN                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   flags       |   loop-count  |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+          value                |
.                                   (variable length)           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Option-code: OPTION_vendor (G128~159)

Option-len: The total length in octets.

PEN: Private Enterprise Number.

flags: See Section 3.8.1

loop-count: See Section 3.8.1 This field is present if and only if the objective is a negotiation 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.

3.8.5. Experimental Objective Options

Option codes G176~191 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. Open Issues

There are various unresolved design questions that are worthy of more work in the near future, as listed below (statically numbered in historical order for reference purposes, with the resolved issues retained for reference):

o 1. UDP vs TCP: For now, this specification suggests UDP and TCP as message transport mechanisms. This is not clarified yet. UDP is good for short conversations, is necessary for multicast discovery, and generally fits the discovery and divert scenarios well. However, it will cause problems with large messages. TCP is good for stable and long sessions, with a little bit of time consumption during the session establishment stage. If messages exceed a reasonable MTU, a TCP mode will be required in any case. This question may be affected by the security discussion.

  RESOLVED by specifying UDP for short message and TCP for longer one.

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

  RESOLVED by specifying external security (ACP or (D)TLS).

- o The following open issues 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 states that this option could be any place in a message. Wouldn’t it be better to specify a position (such as the end)? That would be much simpler to implement.

  RESOLVED by changing security model.

- o 3. DoS Attack Protection needs work.

  RESOLVED by adding text.

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

RESOLVED for now by adding optional use of DNS-SD by ASAs.

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

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

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

- o 8. Consideration of ADNCP proposal.

  RESOLVED by adding optional use of ADNCP for flooding-type synchronization.

- o 9. Clarify how a GDNP instance knows whether it is running inside the ACP. (Sheng)

  RESOLVED by improved text.

- o 10. Clarify how a non-ACP GDNP instance initiates (D)TLS. (Sheng)

  RESOLVED by improved text and declaring DTLS out of scope for this draft.

- o 11. Clarify how UDP/TCP choice is made. (Sheng) [Like DNS? - Brian]

  RESOLVED by improved text.

- o 12. Justify that IP address within ACP or (D)TLS environment is sufficient to prove AN identity; or explain how Device Identity Option is used. (Sheng)

RESOLVED for now: we assume that all ASAs in a device are trusted as soon as the device is trusted, so they share credentials. In that case the Device Identity Option is useless. This needs to be reviewed later.

- o 13. Emphasise that negotiation/synchronization are independent from discovery, although the rapid discovery mode includes the first step of a negotiation/synchronization. (Sheng)

  RESOLVED by improved text.

- o 14. Do we need an unsolicited flooding mechanism for discovery (for discovery results that everyone needs), to reduce scaling impact of flooding discovery messages? (Toerless)

  RESOLVED: Yes, added to requirements and solution.

- o 15. Do we need flag bits in Objective Options to distinguish distinguish Synchronization and Negotiation "Request" or rapid mode "Discovery" messages? (Bing)

  RESOLVED: yes, work on the API showed that these flags are essential.

- o 16. (Related to issue 14). Should we revive the "unsolicited Response" for flooding synchronisation data? This has to be done carefully due to the well-known issues with flooding, but it could be useful, e.g. for Intent distribution, where DNCP doesn’t seem applicable.

- o 17. Ensure that the discovery mechanism is completely proof against loops and protected against duplicate responses.

- o 18. Discuss the handling of multiple valid discovery responses.

- o 19. Should we use a text-oriented format such as JSON/CBOR instead of native binary TLV format?

- o 20. Is the Divert option needed? If a discovery response provides a valid IP address or FQDN, the recipient doesn’t gain any extra knowledge from the Divert.

- o 21. Rename the protocol as GRASP (GeneRic Autonomic Signaling Protocol)?

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 MUST be capable of proving its identity and authenticating its messages. GDNP relies on a separate certificate-based security mechanism to support authentication, data integrity protection, and anti-replay protection.

  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. In a network requiring "air gap" security, such a dependency would be unacceptable.

- Privacy and confidentiality

  Generally speaking, no personal information is expected to be involved in the signaling 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

  GDNP discovery partly relies on insecure link-local multicast. Since routers participating in GDNP sometimes relay discovery messages from one link to another, this could be a vector for denial of service attacks. Relevant mitigations are specified in Section 3.3.3. Additionally, it is of great importance that firewalls prevent any GDNP messages from entering the domain from an untrusted source.

- 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 succesfully 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 link-local multicast addresses, which have been assigned by IANA for use by GDNP:

ALL_GDNP_NEIGHBOR multicast address (IPv6): (TBD1). Assigned in the IPv6 Link-Local Scope Multicast Addresses registry.

ALL_GDNP_NEIGHBOR multicast address (IPv4): (TBD2). Assigned in the IPv4 Multicast Local Network Control Block.

(Note in draft: alternatively, we could use 224.0.0.1, currently defined as All Systems on this Subnet.)

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 defines the General Discovery and Negotiation Protocol (GDNP). The IANA is requested to create a GDNP registry within the unused portion of the DNCP registry [I-D.ietf-homenet-dncp]. 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.

Note to the RFC Editor: In the following tables and in the body of this document, the values G0, G1, etc., should be replaced by the assigned values.

GDNP Messages table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in Section 3.6 in this document:

Type	Name	RFCs
G0	Reserved	this document
G1	Discovery Message	this document
G2	Response Message	this document
G3	Request Message	this document
G4	Negotiation Message	this document
G5	Negotiation-ending Message	this document
G6	Confirm-waiting Message	this document
G7~31	reserved for future messages	

GDNP Options table. The values in this table are 16-bit unsigned integers. The following initial values are assigned in Section 3.7 and Section 3.8.1 in this document:

Type	Name	RFCs
G32	Divert Option	this document
G33	Accept Option	this document
G34	Decline Option	this document
G35	Waiting Time Option	this document
G36	Device Identity Option	this document
G37	Locator IPv4 Address Option	this document
G38	Locator IPv6 Address Option	this document
G39	Locator FQDN Option	this document
G40~63	Reserved for future GDNP General Options	
G64~127	Reserved for future GDNP Objective Options	
G128~159	Vendor Specific Options	this document
G160~175	Reserved for future use	
G176~191	Experimental Options	this document
G192~???	Reserved for future use	

7. Acknowledgements

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

Valuable comments were received from Michael Behringer, Jeferson Campos Nobre, Laurent Ciavaglia, 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-04, 2015-06-21:

Tuned wording around hierarchical structure.

Changed "device" to "ASA" in many places.

Reformulated requirements to be clear that the ASA is the main customer for signaling.

Added requirement for flooding unsolicited synch, and added it to protocol spec. Recognized DNCP as alternative for flooding synch data.

Requirements clarified, expanded and rearranged following design team discussion.

Clarified that GDNP discovery must not be a prerequisite for GDNP negotiation or synchronization (resolved issue 13).

Specified flag bits for objective options (resolved issue 15).

Clarified usage of ACP vs TLS/DTLS and TCP vs UDP (resolved issues 9,10,11).

Updated DNCP description from latest DNCP draft.

Editorial improvements.

draft-carpenter-anima-discovery-negotiation-protocol-03, 2015-04-20:

Removed intrinsic security, required external security

Format changes to allow ADNCP co-existence

Recognized DNS-SD as alternative discovery method.

Editorial improvements

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 Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

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-02 (work in progress), March 2015.

[I-D.behringer-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., and B. Liu, "A Reference Model for Autonomic Networking", draft-behringer-anima-reference-model-02 (work in progress), June 2015.

[I-D.chaparadza-intarea-igcp]
Behringer, M., Chaparadza, R., Petre, R., Li, X., and H. Mahkonen, "IP based Generic Control Protocol (IGCP)", draft-chaparadza-intarea-igcp-00 (work in progress), July 2011.

[I-D.eckert-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-eckert-anima-stable-connectivity-01 (work in progress), March 2015.

[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-06 (work in progress), March 2015.

[I-D.ietf-homenet-dncp]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", draft-ietf-homenet-dncp-05 (work in progress), June 2015.

[I-D.ietf-homenet-hncp]
Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", draft-ietf-homenet-hncp-06 (work in progress), June 2015.

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-05 (work in progress), June 2015.

[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-05 (work in progress), March 2015.

[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.

[I-D.stenberg-anima-adncp]
Stenberg, M., "Autonomic Distributed Node Consensus Protocol", draft-stenberg-anima-adncp-00 (work in progress), March 2015.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2608] Guttman, E., Perkins, C., Veizades, J., and M. Day, "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 Tunnels", RFC 3209, December 2001.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[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.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, October 2010.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, March 2011.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6733] Fajardo, V., Arkko, J., Loughney, J., and G. Zorn, "Diameter Base Protocol", RFC 6733, October 2012.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, June 2015.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, June 2015.

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. Both SLP and DNS-SD are text-based protocols.

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 some signaling protocols with an element of negotiation. For example Resource ReSerVation Protocol (RSVP) [RFC2205] was designed for negotiating quality of service parameters along the path of a unicast or multicast flow. RSVP is a very specialised protocol aimed at end-to-end flows. 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 signaling across many hops rather than at device-to-device signaling. However, we cannot completely exclude extended RSVP or GIST as a synchronization and negotiation protocol. They do not appear to be directly useable for peer discovery.

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 Home Networking Control Protocol (HNCP) [I-D.ietf-homenet-hncp] for configuring Homenet routers. A specific application of DNCP for autonomic networking was proposed in [I-D.stenberg-anima-adncp].

DNCP "is designed to provide a way for each participating node to publish a set of TLV (Type-Length-Value) tuples, and to provide a shared and common view about the data published... DNCP is most suitable for data that changes only infrequently... If constant rapid state changes are needed, the preferable choice is to use an additional point-to-point channel..."

Specific features of DNCP include:

- 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 is used only for discovery of DNCP neighbors when lower security is acceptable.

- Synchronization of state is maintained by a flooding process using the Trickle algorithm. There is no bilateral synchronization or 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.

DNCP does not meet the needs of a general negotiation protocol, because it is designed specifically for flooding synchronization. Also, in its HNCP profile it is limited to link-local messages and to IPv6. However, at the minimum it is a very interesting test case for this style of interaction between devices without needing a central authority, and it is a proven method of network-wide state synchronization by flooding.

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

A. He, Ed.
Huawei
B. Sarikaya, Ed.
Huawei USA

March 9, 2015

IoT Security Bootstrapping: Survey and Design Considerations
draft-he-6lo-analysis-iot-sbootstrapping-00

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.

Table of Contents

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-zerotouch] 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.

Referenced solution	Authentication method	Credential material
[I-D.pritikin-anima-bootstrapping-keyinfra]	802.1x-EAPOL, EAP-TLS, EAP-IKEv2	802.1AR certificate
[I-D.sarikaya-6lo-bootstrapping-solution]	EAP-TLS (modified)	Raw public key
[I-D.struik-6tisch-security-considerations]	Joining Protocol (undefined)	Certificate
[I-D.kwatsen-netconf-zerotouch]	Unspecified (EAP-TLS might be used)	X.509 certificate
[I-D.he-iot-security-bootstrapping]	EAP, PANA	Unspecified
[I-D.kumar-6lo-selective-bootstrap]	Selected by Commissioner with CT	PSK defined in CT
ZigBee IP stack based Smart Energy	EAP-TLS, PANA	Certificate
Thread networking	Unknown	Product install codes

Table 1

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

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.

- o 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.

- o 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.

- o 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]
IEEE Standard, , "IEEE Std. 802.15.4-2011", October 2011, <http://standards.ieee.org/findstds/standard/802.15.4-2011.html>.

[IEEE802.15.9]
IEEE P802.15.9/D01, "IEEE Draft Recommended Practice for transport of key management protocol (KMP) datagrams", November 2014, <http://grouper.ieee.org/groups/802/15/private/members_area.html>.

[IEEE802.1x]
IEEE Std 802.1X-2010, "IEEE 802.1X Port-Based Network Access Control", February 2010, <http://standards.ieee.org/getieee802/download/802.1X-2010.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003.

[RFC3626] Clausen, T. and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)", RFC 3626, October 2003.

[RFC4728] Johnson, D., Hu, Y., and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4", RFC 4728, February 2007.

[RFC4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method", RFC 4764, January 2007.

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

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

[RFC5106] Tschofenig, H., Kroeselberg, D., Pashalidis, A., Ohba, Y., and F. Bersani, "The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method", RFC 5106, February 2008.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.

[RFC5487] Badra, M., "Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode", RFC 5487, March 2009.

[RFC6345] Duffy, P., Chakrabarti, S., Cragie, R., Ohba, Y., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA) Relay Element", RFC 6345, August 2011.

[RFC6550] Winter, T., Thubert, P., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, March 2012.

[RFC6786] Yegin, A. and R. Cragie, "Encrypting the Protocol for Carrying Authentication for Network Access (PANA) Attribute-Value Pairs", RFC 6786, November 2012.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, May 2014.

[RFC7250] Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, June 2014.

[RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", RFC 7251, June 2014.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, June 2014.

[RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7400, November 2014.

[SEP2.0] ZigBee Alliance, "ZigBee IP Specification", March 2014, <hhttp://www.zigbee.org/non-menu-pages/zigbee-ip-download/>.

8.2. Informative References

[I-D.garcia-core-security]
Garcia-Morchon, O., Kumar, S., Keoh, S., Hummen, R., and R. Struik, "Security Considerations in the IP-based Internet of Things", draft-garcia-core-security-06 (work in progress), September 2013.

[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]
Watsen, K., Hanna, S., Clarke, J., and M. Abrahamsson, "Zero Touch Provisioning for NETCONF Call Home (ZeroTouch)", draft-kwatsen-netconf-zerotouch-01 (work in progress), February 2014.

[I-D.nikander-esp-beet-mode]
Nikander, P. and J. Melen, "A Bound End-to-End Tunnel (BEET) mode for ESP", draft-nikander-esp-beet-mode-09 (work in progress), August 2008.

[I-D.oflynn-core-bootstrapping]
Sarikaya, B., Ohba, Y., Cao, Z., and R. Cragie, "Security Bootstrapping of Resource-Constrained Devices", draft-oflynn-core-bootstrapping-03 (work in progress), November 2010.

[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.

[I-D.sarikaya-6lo-bootstrapping-solution]
Sarikaya, B., "Secure Bootstrapping Solution for Resource-Constrained Devices", draft-sarikaya-6lo-bootstrapping-solution-00 (work in progress), June 2013.

[I-D.struik-6tisch-security-considerations]
Struik, R., "6TiSCH Security Architectural Considerations", draft-struik-6tisch-security-considerations-01 (work in progress), January 2015.

[MIT2014] Herder, C., Farinaz Koushanfar, F., and S. Srinivas Devadas, "Physical Unclonable Functions and Applications: A Tutorial", Proceedings of the IEEE , vol. 102, no. 8, pp. 1126-1141, August 2014.

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

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2015

K. Watsen
Juniper Networks
J. Clarke
Cisco Systems
M. Abrahamsson
T-Systems

March 9, 2015

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.

Table of Contents

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

- o 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.

- o 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:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o 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.

```
            manufactures
         +-----------------------------------+
         |                                   |
         |              1.discover           |
         |                bootstrap          |   2.fetch bootstrap
         |                information       v     data and provide
         |                (optional)    +------+  confirmation
         |              +---------------|Device|----------------+
         |              |               +------+                |
         |              |                   |                   |
         |              v                   |                   v
      +------+       +------+               |              +---------+
      |Vendor|       | DHCP |               |3.netconf     |Bootstrap|
      +------+       |Server|               | callhome     | Server  |
        |  ^         +------+               |              +---------+
        |  |            ^                   |                   ^
        |  |            | stage             |                   |
        |  |            | bootstrap         |      stage        |
        |  |            | information       v      bootstrap    |
        |  |            | (optional)    +-------+  data        |
        |  |            +---------------|  NMS  |---------------+
        |  |                            +-------+
        |  |  imports trust anchor,        | ^
        |  |  signups for owner cert,      | |
        |  |  and orders devices           | |
        |  +-------------------------------+ |
       +-------------------------------------+
         ships devices, provides
         serial-numbers and/or IDevID
         certs, and ownership vouchers
```

The boxes in this diagram are described next. A sequence diagram explaining the various calls follows in Section 2.2.

o 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.

o 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.

o 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.

o 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.

- o 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.

```
+------+        +------+         +---+        +---------+       +------+
|Vendor|        |Device|         |NMS|        |Bootstrap|       | DHCP |
+------+        +------+         +---+        | Server  |       |Server|
   |               |               |          +---------+       +------+
   |               |               |               |               |
   |  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      |               |
   x               |               |    bootstrap  |               |
                   |               |    data       |               |
                   |               |-------------->|               |
                   |               |               |               |
                   |               |  7. stage bootstrap           |
                   |               |     info (optional)           |
                   |               |------------------------------>|
     8. power on   |               |               |               |
    -------------->|               |               |               |
                   | 9. get networking settings and|               |
                   |    staged bootstrap info (if any)             |
                   |----------------------------------------------->|
                   |               |               |               |
                   |               |               |               x
                   | 10. update boot-image, if     |
                   |     needed, and install       |
                   |     config, if valid          |
                   |------------------------------>|
                   |               |               |
                   |               |               x
                   | 11. netconf   |
                   |     call-home |
                   |+------------->|
                   |               |
                   v               v
```

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 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 digram 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 {

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server";
  prefix "ztbs";

  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 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.";
        }
      }

      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";
      }
    }
  }

}

<CODE ENDS>
```

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

```
+-----------------------------------------------------------------+
|                            <device>                             |
|                                                                 |
|   +-------------------------------------------------------+     |
|   |                  <read-only storage>                  |     |
|   |                                                       |     |
|   | 1. list of public Internet Bootstrap Servers          |     |
|   | 2. list of trust anchor certs for Bootstrap Servers   |     |
|   | 3. trust anchor cert for owner certificates           |     |
|   | 4. trust anchor cert for device ownership vouchers    |     |
|   | 5. IDevID cert & associated intermediate certificate(s)|     |
|   +-------------------------------------------------------+     |
|                                                                 |
|                    +----------------------+                     |
|                    |   <secure storage>   |                     |
|                    |                      |                     |
|                    |  6. private key      |                     |
|                    +----------------------+                     |
|                                                                 |
+-----------------------------------------------------------------+
```

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

```
+-----------------------------------------------------------------+
|                              <nms>                              |
|                                                                 |
|     +-------------------------------------------------------+   |
|     |                 <read-only storage>                   |   |
|     |                                                       |   |
|     |  1. list of IDevID trust anchor certificates          |   |
|     +-------------------------------------------------------+   |
|                                                                 |
|     +-------------------------------------------------------+   |
|     |                   <configuration>                     |   |
|     |                                                       |   |
|     |  2. list of expected device unique identifiers        |   |
|     +-------------------------------------------------------+   |
|                                                                 |
|     +-------------------------------------------------------+   |
|     |                  <secure storage>                     |   |
|     |                                                       |   |
|     |  3. map of device identifiers to login credentials    |   |
|     +-------------------------------------------------------+   |
|                                                                 |
+-----------------------------------------------------------------+
```

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.

```
 Code   Len
+-----+-----+------+------+----
| XXX |  n  | svr1 | svr2 | ...
+-----+-----+------+------+----
```

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.

```
 Code   Len
+-----+-----+------+------+----
| YYY |  n  | svr1 | svr2 | ...
+-----+-----+------+------+----
```

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, March 2011.

[Std-802.1AR-2009]
IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[draft-ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home (work in progress)", October 2014, <https://tools.ietf.org/html/draft-ietf-netconf-call-home-04>.

[draft-ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ieft-netconf-restconf-04 (work in progress), 2014, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-04>.

[draft-ietf-netconf-server-model]
Watsen, K., "NETCONF Server Model (work in progress)", September 2014, <http://tools.ietf.org/html/draft-ietf-netconf-server-model-06>.

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 of scope for this draft.

A tree digram 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 {

  namespace "urn:ietf:params:xml:ns:yang: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;
      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
<?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>AAAAB3NzaC1yc2EAAAADAQABAAABAQDeJMV8zrtsi8CgEsRC
          jCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/O2Mwj
          E1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcC
          WAw1lOr9IDGDAuww6G45gLcHalHMmBtQxKnZdzU9kx/fL3ZS5G76Fy6sA5
          vg7SLqQFPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2DKqBIe340jWq
          EIuA7LvEJYql4unq4Iog+/+CiumTkmQIWRgIoj4FCzYkO9NvRE6fOSLLf6
          gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</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>
          <host-keys>
            <host-key>my-call-home-x509-key</host-key>
          </host-keys>
        </ssh>
      </application>
    </call-home>
  </netconf-server>

</configuration>
```

Appendix B. Change Log

B.1. ID to 00

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

B.2. 00 to 01

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

B.3. 01 to 02

- o 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.

- o Renamed Configuration Server to Bootstrap Server, a more representative name given the information devices download from it.

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

- o 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

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 19, 2019

K. Watsen
Juniper Networks
M. Abrahamsson
T-Systems
I. Farrer
Deutsche Telekom AG
January 15, 2019

Secure Zero Touch Provisioning (SZTP)
draft-ietf-netconf-zerotouch-29

Abstract

This draft presents a technique to securely provision a networking device when it is booting in a factory-default state. Variations in the solution enables it to be used on both public and private networks. The provisioning steps are able to update the boot image, commit an initial configuration, and execute arbitrary scripts to address auxiliary needs. The updated device is subsequently able to establish secure connections with other systems. For instance, a device may establish NETCONF (RFC 6241) and/or RESTCONF (RFC 8040) connections with deployment-specific network management systems.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in the IANA Considerations section contains placeholder values for DHCP options pending IANA assignment. Please apply the following replacements:

- "TBD1" --> the assigned value for id-ct-sztpConveyedInfoXML

- "TBD2" --> the assigned value for id-ct-sztpConveyedInfoJSON

- "TBD_IANA_URL" --> the assigned URL for the IANA registry

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

- "XXXX" --> the assigned numerical RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

- o "2019-01-15" --> the publication date of this draft

The following one Appendix section is to be removed prior to publication:

- o Appendix D. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

A fundamental business requirement for any network operator 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 for installations is both cost prohibitive and does not scale.

This document defines Secure Zero Touch Provisioning (SZTP), a bootstrapping strategy enabling devices to securely obtain bootstrapping data with no installer action beyond physical placement and connecting network and power cables. As such, SZTP enables non-technical personnel to bring up devices in remote locations without the need for any operator input.

The SZTP solution includes updating the boot image, committing an initial configuration, and executing arbitrary scripts to address auxiliary needs. The updated device is subsequently able to establish secure connections with other systems. For instance, a devices may establish NETCONF [RFC8040] and/or RESTCONF [RFC6241] connections with deployment-specific network management systems.

This document primarily regards physical devices, where the setting of the device's initial state, described in Section 5.1, occurs during the device's manufacturing process. The SZTP solution may be extended to support virtual machines or other such logical constructs, but details for how this can be accomplished is left for future work.

1.1. Use Cases

- o Device connecting to a remotely administered network

  This use-case involves scenarios, such as a remote branch office or convenience store, whereby a device connects as an access gateway to an ISP's network. Assuming it is not possible to customize the ISP's network to provide any bootstrapping support, and with no other nearby device to leverage, the device has no recourse but to reach out to an Internet-based bootstrap server to bootstrap from.

o Device connecting to a locally administered network

This use-case covers all other scenarios and differs only in that the device may additionally leverage nearby devices, which may direct it to use a local service to bootstrap from. If no such information is available, or the device is unable to use the information provided, it can then reach out to the network just as it would for the remotely administered network use-case.

Conceptual workflows for how SZTP might be deployed are provided in Appendix C.

1.2. Terminology

This document uses the following terms (sorted by name):

Artifact: The term "artifact" is used throughout to represent any of the three artifacts defined in Section 3 (conveyed information, ownership voucher, and owner certificate). These artifacts collectively provide all the bootstrapping data a device may use.

Bootstrapping Data: The term "bootstrapping data" is used throughout this document to refer to the collection of data that a device may obtain during the bootstrapping process. Specifically, it refers to the three artifacts conveyed information, owner certificate, and ownership voucher, as described in Section 3.

Bootstrap Server: The term "bootstrap server" is used within this document to mean any RESTCONF server implementing the YANG module defined in Section 7.3.

Conveyed Information: The term "conveyed information" is used herein to refer either redirect information or onboarding information. Conveyed information is one of the three bootstrapping artifacts described in Section 3.

Device: The term "device" is used throughout this document to refer to a network element that needs to be bootstrapped. See Section 5 for more information about devices.

Manufacturer: The term "manufacturer" is used herein to refer to the manufacturer of a device or a delegate of the manufacturer.

Network Management System (NMS): The acronym "NMS" is used throughout this document to refer to the deployment-specific management system that the bootstrapping process is responsible for introducing devices to. From a device’s perspective, when

the bootstrapping process has completed, the NMS is a NETCONF or RESTCONF client.

Onboarding Information: The term "onboarding information" is used herein to refer to one of the two types of "conveyed information" defined in this document, the other being "redirect information". Onboarding information is formally defined by the "onboarding-information" YANG-data structure in Section 6.3.

Onboarding Server: The term "onboarding server" is used herein to refer to a bootstrap server that only returns onboarding information.

Owner: The term "owner" is used throughout this document to refer to the person or organization that purchased or otherwise owns a device.

Owner Certificate: The term "owner certificate" is used in this document to represent an X.509 certificate that binds an owner identity to a public key, which a device can use to validate a signature over the conveyed information artifact. The owner certificate may be communicated along with its chain of intermediate certificates leading up to a known trust anchor. The owner certificate is one of the three bootstrapping artifacts described in Section 3.

Ownership Voucher: The term "ownership voucher" is used in this document to represent the voucher artifact defined in [RFC8366]. The ownership voucher is used to assign a device to an owner. The ownership voucher is one of the three bootstrapping artifacts described in Section 3.

Redirect Information: The term "redirect information" is used herein to refer to one of the two types of "conveyed information" defined in this document, the other being "onboarding information". Redirect information is formally defined by the "redirect-information" YANG-data structure in Section 6.3.

Redirect Server: The term "redirect server" is used to refer to a bootstrap server that only returns redirect information. A redirect server is particularly useful when hosted by a manufacturer, as a well-known (e.g., Internet-based) resource to redirect devices to deployment-specific bootstrap servers.

Signed Data: The term "signed data" is used throughout to mean conveyed information that has been signed, specifically by a private key possessed by a device’s owner.

Unsigned Data: The term "unsigned data" is used throughout to mean conveyed information that has not been signed.

1.3. Requirements Language

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

1.4. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. Types of Conveyed Information

This document defines two types of conveyed information that devices can access during the bootstrapping process. These conveyed information types are described in this section. Examples are provided in Section 6.2

2.1. Redirect Information

Redirect information redirects a device to another bootstrap server. Redirect information encodes a list of bootstrap servers, each specifying the bootstrap server's hostname (or IP address), an optional port, and an optional trust anchor certificate that the device can use to authenticate the bootstrap server with.

Redirect information is YANG modeled data formally defined by the "redirect-information" container in the YANG module presented in Section 6.3. This container has the tree diagram shown below.

```
+--:(redirect-information)
   +-- redirect-information
      +-- bootstrap-server* [address]
         +-- address         inet:host
         +-- port?           inet:port-number
         +-- trust-anchor?   cms
```

Redirect information may be trusted or untrusted. The redirect information is trusted whenever it is obtained via a secure connection to a trusted bootstrap server, or whenever it is signed by the device's owner. In all other cases, the redirect information is untrusted.

Trusted redirect information is useful for enabling a device to establish a secure connection to a specified bootstrap server, which is possible when the redirect information includes the bootstrap server's trust anchor certificate.

Untrusted redirect information is useful for directing a device to a bootstrap server where signed data has been staged for it to obtain. Note that, when the redirect information is untrusted, devices discard any potentially included trust anchor certificates.

How devices process redirect information is described in Section 5.5.

2.2. Onboarding Information

Onboarding information provides data necessary for a device to bootstrap itself and establish secure connections with other systems. As defined in this document, onboarding information can specify details about the boot image a device must be running, specify an initial configuration the device must commit, and specify scripts that the device must successfully execute.

Onboarding information is YANG modeled data formally defined by the "onboarding-information" container in the YANG module presented in Section 6.3. This container has the tree diagram shown below.

```
+--:(onboarding-information)
   +-- onboarding-information
      +-- boot-image
      |  +-- os-name?              string
      |  +-- os-version?           string
      |  +-- download-uri*         inet:uri
      |  +-- image-verification* [hash-algorithm]
      |     +-- hash-algorithm    identityref
      |     +-- hash-value        yang:hex-string
      +-- configuration-handling?      enumeration
      +-- pre-configuration-script?    script
      +-- configuration?               binary
      +-- post-configuration-script?   script
```

Onboarding information must be trusted for it to be of any use to a device. There is no option for a device to process untrusted onboarding information.

Onboarding information is trusted whenever it is obtained via a secure connection to a trusted bootstrap server, or whenever it is signed by the device's owner. In all other cases, the onboarding information is untrusted.

How devices process onboarding information is described in Section 5.6.

3. Artifacts

This document defines three artifacts that can be made available to devices while they are bootstrapping. Each source of bootstrapping data specifies how it provides the artifacts defined in this section (see Section 4).

3.1. Conveyed Information

The conveyed information artifact encodes the essential bootstrapping data for the device. This artifact is used to encode the redirect information and onboarding information types discussed in Section 2.

The conveyed information artifact is a CMS structure, as described in [RFC5652], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690 [ITU.X690.2015]. The CMS structure MUST contain content conforming to the YANG module specified in Section 6.3.

The conveyed information CMS structure may encode signed or unsigned bootstrapping data. When the bootstrapping data is signed, it may also be encrypted but, from a terminology perspective, it is still "signed data" Section 1.2.

When the conveyed information artifact is unsigned, as it might be when communicated over trusted channels, the CMS structure's top-most content type MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing "conveyed-information" data in the expected encoding.

When the conveyed information artifact is unsigned and encrypted, as it might be when communicated over trusted channels but, for some reason, the operator wants to ensure that only the device is able to see the contents, the CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3). Furthermore, the encryptedContentInfo's content type MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing "conveyed-information" data in the expected encoding.

When the conveyed information artifact is signed, as it might be when communicated over untrusted channels, the CMS structure's top-most content type MUST be the OID id-signedData (1.2.840.113549.1.7.2). Furthermore, the inner eContentType MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content or eContent is an octet string containing "conveyed-information" data in the expected encoding.

When the conveyed information artifact is signed and encrypted, as it might be when communicated over untrusted channels and privacy is important, the CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3). Furthermore, the encryptedContentInfo's content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whose eContentType MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content or eContent is an octet string containing "conveyed-information" data in the expected encoding.

3.2. Owner Certificate

The owner certificate artifact is an X.509 certificate [RFC5280] that is used to identify an "owner" (e.g., an organization). The owner certificate can be signed by any certificate authority (CA). The owner certificate either MUST have no Key Usage specified or the Key Usage MUST at least set the "digitalSignature" bit. The values for the owner certificate's "subject" and/or "subjectAltName" are not constrained by this document.

The owner certificate is used by a device to verify the signature over the conveyed information artifact (Section 3.1) that the device should have also received, as described in Section 3.5. In particular, the device verifies the signature using the public key in the owner certificate over the content contained within the conveyed information artifact.

The owner certificate artifact is formally a CMS structure, as specified by [RFC5652], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690 [ITU.X690.2015].

The owner certificate CMS structure MUST contain the owner certificate itself, as well as all intermediate certificates leading to the "pinned-domain-cert" certificate specified in the ownership

voucher. The owner certificate artifact MAY optionally include the "pinned-domain-cert" as well.

In order to support devices deployed on private networks, the owner certificate CMS structure MAY also contain suitably fresh, as determined by local policy, revocation objects (e.g., CRLs). Having these revocation objects stapled to the owner certificate may obviate the need for the device to have to download them dynamically using the CRL distribution point or an OCSP responder specified in the associated certificates.

When unencrypted, the owner certificate artifact's CMS structure's top-most content type MUST be the OID id-signedData (1.2.840.113549.1.7.2). The inner SignedData structure is the degenerate form, whereby there are no signers, that is commonly used to disseminate certificates and revocation objects.

When encrypted, the owner certificate artifact's CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3), and the encryptedContentInfo's content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whereby the inner SignedData structure is the degenerate form that has no signers commonly used to disseminate certificates and revocation objects.

3.3. Ownership Voucher

The ownership voucher artifact is used to securely identify a device's owner, as it is known to the manufacturer. The ownership voucher is signed by the device's manufacturer.

The ownership voucher is used to verify the owner certificate (Section 3.2) that the device should have also received, as described in Section 3.5. In particular, the device verifies that the owner certificate has a chain of trust leading to the trusted certificate included in the ownership voucher ("pinned-domain-cert"). Note that this relationship holds even when the owner certificate is a self-signed certificate, and hence also the pinned-domain-cert.

When unencrypted, the ownership voucher artifact is as defined in [RFC8366]. As described, it is a CMS structure whose top-most content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whose eContentType MUST be OID id-ct-animaJSONVoucher (1.2.840.113549.1.9.16.1), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing ietf-voucher data in the expected encoding.

When encrypted, the ownership voucher artifact's CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3), and the encryptedContentInfo's content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whose eContentType MUST be OID id-ct-animaJSONVoucher (1.2.840.113549.1.9.16.1), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content is an octet string containing ietf-voucher data in the expected encoding.

3.4. Artifact Encryption

Each of the three artifacts MAY be individually encrypted. Encryption may be important in some environments where the content is considered sensitive.

Each of the three artifacts are encrypted in the same way, by the unencrypted form being encapsulated inside a CMS EnvelopedData type.

As a consequence, both the conveyed information and ownership voucher artifacts are signed and then encrypted, never encrypted and then signed.

This sequencing has the advantage of shrouding the signer's certificate, and ensuring that the owner knows the content being signed. This sequencing further enables the owner to inspect an unencrypted voucher obtained from a manufacturer and then encrypt the voucher later themselves, perhaps while also stapling in current revocation objects, when ready to place the artifact in an unsafe location.

When encrypted, the CMS MUST be encrypted using a secure device identity certificate for the device. This certificate MAY be the same as the TLS-level client certificate the device uses when connecting to bootstrap servers. The owner must possess the device's identity certificate at the time of encrypting the data. How the owner comes to posses the device's identity certificate for this purpose is outside the scope of this document.

3.5. Artifact Groupings

The previous sections discussed the bootstrapping artifacts, but only certain groupings of these artifacts make sense to return in the various bootstrapping situations described in this document. These groupings are:

Unsigned Data: This artifact grouping is useful for cases when transport level security can be used to convey trust (e.g., HTTPS), or when the conveyed information can be processed in a provisional manner (i.e. unsigned redirect information).

Signed Data, without revocations: This artifact grouping is useful when signed data is needed (i.e., because the data is obtained from an untrusted source and it cannot be processed provisionally) and either revocations are not needed or the revocations can be obtained dynamically.

Signed Data, with revocations: This artifact grouping is useful when signed data is needed (i.e., because the data is obtained from an untrusted source and it cannot be processed provisionally), and revocations are needed, and the revocations cannot be obtained dynamically.

The presence of each artifact, and any distinguishing characteristics, are identified for each artifact grouping in the table below ("yes/no" regards if the artifact is present in the artifact grouping):

Artifact Grouping	Conveyed Information	Ownership Voucher	Owner Certificate
Unsigned Data	Yes, no sig	No	No
Signed Data, without revocations	Yes, with sig	Yes, without revocations	Yes, without revocations
Signed Data, with revocations	Yes, with sig	Yes, with revocations	Yes, with revocations

4. Sources of Bootstrapping Data

This section defines some sources for bootstrapping data that a device can access. The list of sources defined here is not meant to be exhaustive. It is left to future documents to define additional sources for obtaining bootstrapping data.

For each source of bootstrapping data defined in this section, details are given for how the three artifacts listed in Section 3 are provided.

4.1. Removable Storage

A directly attached removable storage device (e.g., a USB flash drive) MAY be used as a source of SZTP bootstrapping data.

Use of a removable storage device is compelling, as it does not require any external infrastructure to work. It is notable that the raw boot image file can also be located on the removable storage device, enabling a removable storage device to be a fully self-standing bootstrapping solution.

To use a removable storage device as a source of bootstrapping data, a device need only detect if the removable storage device is plugged in and mount its filesystem.

A removable storage device is an untrusted source of bootstrapping data. This means that the information stored on the removable storage device either MUST be signed or MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a removable storage device presents itself as a filesystem, the bootstrapping artifacts need to be presented as files. The three artifacts defined in Section 3 are mapped to files below.

Artifact to File Mapping:

Conveyed Information: Mapped to a file containing the binary artifact described in Section 3.1 (e.g., conveyed-information.cms).

Owner Certificate: Mapped to a file containing the binary artifact described in Section 3.2 (e.g., owner-certificate.cms).

Ownership Voucher: Mapped to a file containing the binary artifact described in Section 3.3 (e.g., ownership-voucher.cms or ownership-voucher.vcj).

The format of the removable storage device’s filesystem and the naming of the files are outside the scope of this document. However, in order to facilitate interoperability, it is RECOMMENDED devices support open and/or standards based filesystems. It is also RECOMMENDED that devices assume a file naming convention that enables more than one instance of bootstrapping data (i.e., for different devices) to exist on a removable storage device. The file naming convention SHOULD additionally be unique to the manufacturer, in

order to enable bootstrapping data from multiple manufacturers to exist on a removable storage device.

4.2. DNS Server

A DNS server MAY be used as a source of SZTP bootstrapping data.

Using a DNS server may be a compelling option for deployments having existing DNS infrastructure, as it enables a touchless bootstrapping option that does not entail utilizing an Internet based resource hosted by a 3rd-party.

DNS is an untrusted source of bootstrapping data. Even if DNSSEC [RFC6698] is used to authenticate the various DNS resource records (e.g., A, AAAA, CERT, TXT, and TLSA), the device cannot be sure that the domain returned to it from e.g., a DHCP server, belongs to its rightful owner. This means that the information stored in the DNS records either MUST be signed (per this document, not DNSSEC), or MUST be information that can be processed provisionally (e.g., unsigned redirect information).

4.2.1. DNS Queries

Devices claiming to support DNS as a source of bootstrapping data MUST first query for device-specific DNS records and, only if doing so does not result in a successful bootstrap, then MUST query for device-independent DNS records.

For each of the device-specific and device-independent queries, devices MUST first query using multicast DNS [RFC6762] and, only if doing so does not result in a successful bootstrap, then MUST query again using unicast DNS [RFC1035] [RFC7766], assuming the address of a DNS server is known, such as it may be using techniques similar to those described in Section 11 of [RFC6763], which is referenced a few times in this document, even though this document does not itself use DNS-SD (RFC 6763 is identified herein as an Informative reference).

When querying for device-specific DNS records, devices MUST query for TXT records [RFC1035] under "<serial-number>._sztp", where <serial-number> is the device’s serial number (the same value as in the device’s secure device identity certificate), and "_sztp" is the globally scoped DNS attribute registered by this document in Section 10.7.

Example device-specific DNS record queries:

```
TXT in <serial-number>._sztp.local.  (multicast)
TXT in <serial-number>._sztp.<domain>.  (unicast)
```

When querying for device-independent DNS records, devices MUST query for SRV records [RFC2782] under "_sztp._tcp", where "_sztp" is the service name registered by this document in Section 10.6, and "_tcp" is the globally scoped DNS attribute registered by [I-D.ietf-dnsop-attrleaf].

Note that a device-independent response is anyway only able to encode unsigned data, since signed data necessitates the use of a device-specific ownership voucher. Use of SRV records maximumly leverages existing DNS standards. A response containing multiple SRV records is comparable to an unsigned redirect information's list of bootstrap servers.

Example device-independent DNS record queries:

```
SRV in _sztp._tcp.local.  (multicast)
SRV in _sztp._tcp.<domain>.  (unicast)
```

4.2.2. DNS Response for Device-Specific Queries

For device-specific queries, the three bootstrapping artifacts defined in Section 3 are encoded into the TXT records using key/value pairs, similar to the technique described in Section 6.3 of [RFC6763].

Artifact to TXT Record Mapping:

Conveyed Information: Mapped to a TXT record having the key "ci" and the value being the binary artifact described in Section 3.1.

Owner Certificate: Mapped to a TXT record having the key "oc" and the value being the binary artifact described in Section 3.2.

Ownership Voucher: Mapped to a TXT record having the key "ov" and the value being the binary artifact described in Section 3.3.

Devices MUST ignore any other keys that may be returned.

Note that, despite the name, TXT records can and SHOULD (per Section 6.5 of [RFC6763]) encode binary data.

Following is an example of a device-specific response, as it might be presented by a user-agent, containing signed data. This example assumes that the device's serial number is "<serial-number>", the domain is "example.com", and that "<binary data>" represents the binary artifact:

```
<serial-number>._sztp.example.com. 3600 IN TXT "ci=<binary data>"
<serial-number>._sztp.example.com. 3600 IN TXT "oc=<binary data>"
<serial-number>._sztp.example.com. 3600 IN TXT "ov=<binary data>"
```

Note that, in the case that "ci" encodes unsigned data, the "oc" and "ov" keys would not be present in the response.

4.2.3. DNS Response for Device-Independent Queries

For device-independent queries, the three bootstrapping artifacts defined in Section 3 are encoded into the SVR records as follows.

Artifact to SRV Record Mapping:

Conveyed Information: This artifact is not supported directly. Instead, the essence of unsigned redirect information is mapped to SVR records per [RFC2782].

Owner Certificate: Not supported. Device-independent responses are never encode signed data, and hence there is no need for an owner certificate artifact.

Ownership Voucher: Not supported. Device-independent responses are never encode signed data, and hence there is no need for an ownership voucher artifact.

Following is an example of a device-independent response, as it might be presented by a user-agent, containing (effectively) unsigned redirect information to four bootstrap servers. This example assumes that the domain is "example.com" and that there are four bootstrap servers "sztp[1-4]":

```
_sztp._tcp.example.com. 1800 IN SRV 0 0 443 sztp1.example.com.
_sztp._tcp.example.com. 1800 IN SRV 1 0 443 sztp2.example.com.
_sztp._tcp.example.com. 1800 IN SRV 2 0 443 sztp3.example.com.
_sztp._tcp.example.com. 1800 IN SRV 2 0 443 sztp4.example.com.
```

Note that, in this example, "sztp3" and "sztp4" have equal priority, and hence effectively represent a clustered pair of bootstrap servers. While "sztp1" and "sztp2" only have a single SRV record each, it may be that the record points to a load-balancer fronting a cluster of bootstrap servers.

While this document does not use DNS-SD [RFC6763], per Section 12.2 of that RFC, mDNS responses SHOULD also include all address records (type "A" and "AAAA") named in the SRV rdata.

4.2.4. Size of Signed Data

The signed data artifacts are large by DNS conventions. In the smallest-footprint scenario, they are each a few kilobytes in size. However, onboarding information can easily be several kilobytes in size, and has the potential to be many kilobytes in size.

All resource records, including TXT records, have an upper size limit of 65535 bytes, since "RDLENGTH" is a 16-bit field (Section 3.2.1 in [RFC1035]). If it is ever desired to encode onboarding information that exceeds this limit, the DNS records returned should instead encode redirect information, to direct the device to a bootstrap server from which the onboarding information can be obtained.

Given the expected size of the TXT records, it is unlikely that signed data will fit into a UDP-based DNS packet, even with the EDNS(0) Extensions [RFC6891] enabled. Depending on content, signed data may also not fit into a multicast DNS packet, which bounds the size to 9000 bytes, per Section 17 in [RFC6762]. Thus it is expected that DNS Transport over TCP [RFC7766] will be required in order to return signed data.

4.3. DHCP Server

A DHCP server MAY be used as a source of SZTP bootstrapping data.

Using a DHCP server may be a compelling option for deployments having existing DHCP infrastructure, as it enables a touchless bootstrapping option that does not entail utilizing an Internet based resource hosted by a 3rd-party.

A DHCP server is an untrusted source of bootstrapping data. Thus the information stored on the DHCP server either MUST be signed, or it MUST be information that can be processed provisionally (e.g., unsigned redirect information).

However, unlike other sources of bootstrapping data described in this document, the DHCP protocol (especially DHCP for IPv4) is very limited in the amount of data that can be conveyed, to the extent that signed data cannot be communicated. This means that only unsigned redirect information can be conveyed via DHCP.

Since the redirect information is unsigned, it SHOULD NOT include the optional trust anchor certificate, as it takes up space in the DHCP message, and the device would have to discard it anyway. For this reason, the DHCP options defined in Section 8 do not enable the trust anchor certificate to be encoded.

From an artifact perspective, the three artifacts defined in Section 3 are mapped to the DHCP fields specified in Section 8 as follows.

Artifact to DHCP Option Fields Mapping:

Conveyed Information: This artifact is not supported directly. Instead, the essence of unsigned redirect information is mapped to the DHCP options described in Section 8.

Owner Certificate: Not supported. There is not enough space in the DHCP packet to hold an owner certificate artifact.

Ownership Voucher: Not supported. There is not enough space in the DHCP packet to hold an ownership voucher artifact.

4.4. Bootstrap Server

A bootstrap server MAY be used as a source of SZTP bootstrapping data. A bootstrap server is defined as a RESTCONF [RFC8040] server implementing the YANG module provided in Section 7.

Using a bootstrap server as a source of bootstrapping data is a compelling option as it MAY use transport-level security, obviating the need for signed data, which may be easier to deploy in some situations.

Unlike any other source of bootstrapping data described in this document, a bootstrap server is not only a source of data, but it can also receive data from devices using the YANG-defined "report-progress" RPC defined in the YANG module (Section 7.3). The "report-progress" RPC enables visibility into the bootstrapping process (e.g., warnings and errors), and provides potentially useful information upon completion (e.g., the device’s SSH host-keys).

A bootstrap server may be a trusted or an untrusted source of bootstrapping data, depending on if the device learned about the bootstrap server’s trust anchor from a trusted source. When a bootstrap server is trusted, the conveyed information returned from it MAY be signed. When the bootstrap server is untrusted, the conveyed information either MUST be signed or MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a bootstrap server presents data conforming to a YANG data model, the bootstrapping artifacts need to be mapped to YANG nodes. The three artifacts defined in Section 3

are mapped to "output" nodes of the "get-bootstrapping-data" RPC defined in Section 7.3 below.

Artifact to Bootstrap Server Mapping:

Conveyed Information: Mapped to the "conveyed-information" leaf in the output of the "get-bootstrapping-data" RPC.

Owner Certificate: Mapped to the "owner-certificate" leaf in the output of the "get-bootstrapping-data" RPC.

Ownership Voucher: Mapped to the "ownership-voucher" leaf in the output of the "get-bootstrapping-data" RPC.

SZTP bootstrap servers have only two endpoints, one for the "get-bootstrapping-data" RPC and one for the "report-progress" RPC. These RPCs use the authenticated RESTCONF username to isolate the execution of the RPC from other devices.

5. Device Details

Devices supporting the bootstrapping strategy described in this document MUST have the preconfigured state and bootstrapping logic described in the following sections.

5.1. Initial State

```
+-------------------------------------------------------------+
|                           <device>                          |
|                                                             |
| +-----------------------------------------------------------+
| |                   <read/write storage>                    |
| |                                                           |
| | 1. flag to enable SZTP bootstrapping set to "true"        |
| +-----------------------------------------------------------+
| |                                                           |
| +-----------------------------------------------------------+
| |                   <read-only storage>                     |
| |                                                           |
| | 2. TLS client cert & related intermediate certificates    |
| | 3. list of trusted well-known bootstrap servers           |
| | 4. list of trust anchor certs for bootstrap servers       |
| | 5. list of trust anchor certs for ownership vouchers      |
| +-----------------------------------------------------------+
| |                                                           |
| |   +-------------------------------------------------+     |
| |   |                 <secure storage>                |     |
| |   |                                                 |     |
| |   |  6. private key for TLS client certificate      |     |
| |   |  7. private key for decrypting SZTP artifacts   |     |
| |   +-------------------------------------------------+     |
| |                                                           |
+-------------------------------------------------------------+
```

Each numbered item below corresponds to a numbered item in the diagram above.

1. Devices MUST have a configurable variable that is used to enable/disable SZTP bootstrapping. This variable MUST be enabled by default in order for SZTP bootstrapping to run when the device first powers on. Because it is a goal that the configuration installed by the bootstrapping process disables SZTP bootstrapping, and because the configuration may be merged into the existing configuration, using a configuration node that relies on presence is NOT RECOMMENDED, as it cannot be removed by the merging process.

2. Devices that support loading bootstrapping data from bootstrap servers (see Section 4.4) SHOULD possess a TLS-level client certificate and any intermediate certificates leading to the certificate's well-known trust-anchor. The well-known trust anchor certificate may be an intermediate certificate or a self-signed root certificate. To support devices not having a client certificate, devices MAY, alternatively or in addition to, identify and authenticate themselves to the bootstrap server

using an HTTP authentication scheme, as allowed by Section 2.5 in [RFC8040]; however, this document does not define a mechanism for operator input enabling, for example, the entering of a password.

3. Devices that support loading bootstrapping data from well-known bootstrap servers MUST possess a list of the well-known bootstrap servers. Consistent with redirect information (Section 2.1, each bootstrap server can be identified by its hostname or IP address, and an optional port.

4. Devices that support loading bootstrapping data from well-known bootstrap servers MUST also possess a list of trust anchor certificates that can be used to authenticate the well-known bootstrap servers. For each trust anchor certificate, if it is not itself a self-signed root certificate, the device SHOULD also possess the chain of intermediate certificates leading up to and including the self-signed root certificate.

5. Devices that support loading signed data (see Section 1.2) MUST possess the trust anchor certificates for validating ownership vouchers. For each trust anchor certificate, if it is not itself a self-signed root certificate, the device SHOULD also possess the chain of intermediate certificates leading up to and including the self-signed root certificate.

6. Devices that support using a TLS-level client certificate to identify and authenticate themselves to a bootstrap server MUST possess the private key that corresponds to the public key encoded in the TLS-level client certificate. This private key SHOULD be securely stored, ideally in a cryptographic processor, such as a trusted platform module (TPM) chip.

7. Devices that support decrypting SZTP artifacts MUST posses the private key that corresponds to the public key encoded in the secure device identity certificate used when encrypting the artifacts. This private key SHOULD be securely stored, ideally in a cryptographic processor, such as a trusted platform module (TPM) chip. This private key MAY be the same as the one associated to the TLS-level client certificate used when connecting to bootstrap servers.

A YANG module representing this data is provided in Appendix A.

5.2. Boot Sequence

A device claiming to support the bootstrapping strategy defined in this document MUST support the boot sequence described in this section.

```
        Power On
            |
            v                           No
     1. SZTP bootstrapping configured ------> Boot normally
            |
            | Yes
            v
     2. For each supported source of bootstrapping data,
        try to load bootstrapping data from the source
            |
            |
            v                               Yes
     3. Able to bootstrap from any source? -----> Run with new config
            |
            | No
            v
     4. Loop back to Step 1.


     Note: At any time, the device MAY be configured via an alternate
           provisioning mechanism (e.g., CLI).
```

Each numbered item below corresponds to a numbered item in the diagram above.

1. When the device powers on, it first checks to see if SZTP bootstrapping is configured, as is expected to be the case for the device's preconfigured initial state. If SZTP bootstrapping is not configured, then the device boots normally.

2. The device iterates over its list of sources for bootstrapping data (Section 4). Details for how to processes a source of bootstrapping data are provided in Section 5.3.

3. If the device is able to bootstrap itself from any of the sources of bootstrapping data, it runs with the new bootstrapped configuration.

4. Otherwise the device MUST loop back through the list of bootstrapping sources again.

This document does not limit the simultaneous use of alternate provisioning mechanisms. Such mechanisms may include, for instance, a command line interface (CLI), a web-based user interface, or even another bootstrapping protocol. Regardless how it is configured, the configuration SHOULD unset the flag enabling SZTP bootstrapping discussed in Section 5.1.

5.3. Processing a Source of Bootstrapping Data

This section describes a recursive algorithm that devices can use to, ultimately, obtain onboarding information. The algorithm is recursive because sources of bootstrapping data may return redirect information, which causes the algorithm to run again, for the newly discovered sources of bootstrapping data. An expression that captures all possible successful sequences of bootstrapping data is: zero or more redirect information responses, followed by one onboarding information response.

An important aspect of the algorithm is knowing when data needs to be signed or not. The following figure provides a summary of options:

Kind of Bootstrapping Data		Untrusted Source Can Provide?	Trusted Source Can Provide?
Unsigned Redirect Info	:	Yes+	Yes
Signed Redirect Info	:	Yes	Yes*
Unsigned Onboarding Info	:	No	Yes
Signed Onboarding Info	:	Yes	Yes*

The '+' above denotes that the source redirected to MUST return signed data, or more unsigned redirect information.

The '*' above denotes that, while possible, it is generally unnecessary for a trusted source to return signed data.

The recursive algorithm uses a conceptual global-scoped variable called "trust-state". The trust-state variable is initialized to FALSE. The ultimate goal of this algorithm is for the device to process onboarding information (Section 2.2) while the trust-state variable is TRUE.

If the source of bootstrapping data (Section 4) is a bootstrap server (Section 4.4), and the device is able to authenticate the bootstrap server using X.509 certificate path validation ([RFC6125], Section 6) to one of the device's preconfigured trust anchors, or to a trust anchor that it learned from a previous step, then the device MUST set trust-state to TRUE.

When establishing a connection to a bootstrap server, whether trusted or untrusted, the device MUST identify and authenticate itself to the bootstrap server using a TLS-level client certificate and/or an HTTP authentication scheme, per Section 2.5 in [RFC8040]. If both authentication mechanisms are used, they MUST both identify the same serial number.

When sending a client certificate, the device MUST also send all of the intermediate certificates leading up to, and optionally including, the client certificate's well-known trust anchor certificate.

For any source of bootstrapping data (e.g., Section 4), if any artifact obtained is encrypted, the device MUST first decrypt it using the private key associated with the device certificate used to encrypt the artifact.

If the conveyed information artifact is signed, and the device is able to validate the signed data using the algorithm described in Section 5.4, then the device MUST set trust-state to TRUE; otherwise, if the device is unable to validate the signed data, the device MUST set trust-state to FALSE. Note, this is worded to cover the special case when signed data is returned even from a trusted source of bootstrapping data.

If the conveyed information artifact contains redirect information, the device MUST, within limits of how many recursive loops the device allows, process the redirect information as described in Section 5.5. Implementations MUST limit the maximum number of recursive redirects allowed; the maximum number of recursive redirects allowed SHOULD be no more than ten. This is the recursion step, it will cause the device to reenter this algorithm, but this time the data source will definitely be a bootstrap server, as redirect information is only able to redirect devices to bootstrap servers.

If the conveyed information artifact contains onboarding information, and trust-state is FALSE, the device MUST exit the recursive algorithm (as this is not allowed, see the figure above), returning to the bootstrapping sequence described in Section 5.2. Otherwise, the device MUST attempt to process the onboarding information as described in Section 5.6. Whether the processing of the onboarding information succeeds or fails, the device MUST exit the recursive algorithm, returning to the bootstrapping sequence described in Section 5.2, the only difference being in how it responds to the "Able to bootstrap from any source?" conditional described in the figure in the section.

5.4. Validating Signed Data

Whenever a device is presented signed data, it MUST validate the signed data as described in this section. This includes the case where the signed data is provided by a trusted source.

Whenever there is signed data, the device MUST also be provided an ownership voucher and an owner certificate. How all the needed

artifacts are provided for each source of bootstrapping data is described in Section 4.

In order to validate signed data, the device MUST first authenticate the ownership voucher by validating its signature to one of its preconfigured trust anchors (see Section 5.1), which may entail using additional intermediate certificates attached to the ownership voucher. If the device has an accurate clock, it MUST verify that the ownership voucher was created in the past (i.e., "created-on" < now) and, if the "expires-on" leaf is present, the device MUST verify that the ownership voucher has not yet expired (i.e., now < "expires-on"). The device MUST verify that the ownership voucher's "assertion" value is acceptable (e.g., some devices may only accept the assertion value "verified"). The device MUST verify that the ownership voucher specifies the device's serial number in the "serial-number" leaf. If the "idevid-issuer" leaf is present, the device MUST verify that the value is set correctly. If the authentication of the ownership voucher is successful, the device extracts the "pinned-domain-cert" node, an X.509 certificate, that is needed to verify the owner certificate in the next step.

The device MUST next authenticate the owner certificate by performing X.509 certificate path verification to the trusted certificate extracted from the ownership voucher's "pinned-domain-cert" node. This verification may entail using additional intermediate certificates attached to the owner certificate artifact. If the ownership voucher's "domain-cert-revocation-checks" node's value is set to "true", the device MUST verify the revocation status of the certificate chain used to sign the owner certificate and, if suitably-fresh revocation status is unattainable or if it is determined that a certificate has been revoked, the device MUST NOT validate the owner certificate.

Finally, the device MUST verify that the conveyed information artifact was signed by the validated owner certificate.

If any of these steps fail, the device MUST invalidate the signed data and not perform any subsequent steps.

5.5. Processing Redirect Information

In order to process redirect information (Section 2.1), the device MUST follow the steps presented in this section.

Processing redirect information is straightforward; the device sequentially steps through the list of provided bootstrap servers until it can find one it can bootstrap from.

If a hostname is provided, and the hostname's DNS resolution is to more than one IP address, the device MUST attempt to connect to all of the DNS resolved addresses at least once, before moving on to the next bootstrap server. If the device is able to obtain bootstrapping data from any of the DNS resolved addresses, it MUST immediately process that data, without attempting to connect to any of the other DNS resolved addresses.

If the redirect information is trusted (e.g., trust-state is TRUE), and the bootstrap server entry contains a trust anchor certificate, then the device MUST authenticate the specified bootstrap server's TLS server certificate using X.509 certificate path validation ([RFC6125], Section 6) to the specified trust anchor. If the bootstrap server entry does not contain a trust anchor certificate device, the device MUST establish a provisional connection to the bootstrap server (i.e., by blindly accepting its server certificate), and set trust-state to FALSE.

If the redirect information is untrusted (e.g., trust-state is FALSE), the device MUST discard any trust anchors provided by the redirect information and establish a provisional connection to the bootstrap server (i.e., by blindly accepting its TLS server certificate).

5.6. Processing Onboarding Information

In order to process onboarding information (Section 2.2), the device MUST follow the steps presented in this section.

When processing onboarding information, the device MUST first process the boot image information (if any), then execute the pre-configuration script (if any), then commit the initial configuration (if any), and then execute the post-configuration script (if any), in that order.

When the onboarding information is obtained from a trusted bootstrap server, the device MUST send the "bootstrap-initiated" progress report, and send either a terminating "boot-image-installed-rebooting", "bootstrap-complete", or error specific progress report. If the bootstrap server's "get-bootstrapping-data" RPC-reply's "reporting-level" node is set to "verbose", the device MUST additionally send all appropriate non-terminating progress reports (e.g., initiated, warning, complete, etc.). Regardless of the reporting-level indicated by the bootstrap server, the device MAY send progress reports beyond the mandatory ones specified for the given reporting level.

When the onboarding information is obtained from an untrusted bootstrap server, the device MUST NOT send any progress reports to the bootstrap server, even though the onboarding information was, necessarily, signed and authenticated. Please be aware that bootstrap servers are recommended to promote untrusted connections to trusted connections, in the last paragraph of Section 9.6, so as to, in part, be able to collect progress reports from devices.

If the device encounters an error at any step, it MUST stop processing the onboarding information and return to the bootstrapping sequence described in Section 5.2. In the context of a recursive algorithm, the device MUST return to the enclosing loop, not back to the very beginning. Some state MAY be retained from the bootstrapping process (e.g., updated boot image, logs, remnants from a script, etc.). However, the retained state MUST NOT be active in any way (e.g., no new configuration or running of software), and MUST NOT hinder the ability for the device to continue the bootstrapping sequence (i.e., process onboarding information from another bootstrap server).

At this point, the specific ordered sequence of actions the device MUST perform is described.

If the onboarding information is obtained from a trusted bootstrap server, the device MUST send a "bootstrap-initiated" progress report. It is an error if the device does not receive back the "204 No Content" HTTP status line. If an error occurs, the device MUST try to send a "bootstrap-error" progress report before exiting.

The device MUST parse the provided onboarding information document, to extract values used in subsequent steps. Whether using a stream-based parser or not, if there is an error when parsing the onboarding information, and the device is connected to a trusted bootstrap server, the device MUST try to send a "parsing-error" progress report before exiting.

If boot image criteria are specified, the device MUST first determine if the boot image it is running satisfies the specified boot image criteria. If the device is already running the specified boot image, then it skips the remainder of this step. If the device is not running the specified boot image, then it MUST download, verify, and install, in that order, the specified boot image, and then reboot. If connected to a trusted bootstrap server, the device MAY try to send a "boot-image-mismatch" progress report. To download the boot image, the device MUST only use the URIs supplied by the onboarding information. To verify the boot image, the device MUST either use one of the verification fingerprints supplied by the onboarding information, or use a cryptographic signature embedded into the boot

image itself using a mechanism not described by this document. Before rebooting, if connected to a trusted bootstrap server, the device MUST try to send a "boot-image-installed-rebooting" progress report. Upon rebooting, the bootstrapping process runs again, which will eventually come to this step again, but then the device will be running the specified boot image, and thus will move to processing the next step. If an error occurs at any step while the device is connected to a trusted bootstrap server (i.e., before the reboot), the device MUST try to send a "boot-image-error" progress report before exiting.

If a pre-configuration script has been specified, the device MUST execute the script, capture any output emitted from the script, and check if the script had any warnings or errors. If an error occurs while the device is connected to a trusted bootstrap server, the device MUST try to send a "pre-script-error" progress report before exiting.

If an initial configuration has been specified, the device MUST atomically commit the provided initial configuration, using the approach specified by the "configuration-handling" leaf. If an error occurs while the device is connected to a trusted bootstrap server, the device MUST try to send a "config-error" progress report before exiting.

If a post-configuration script has been specified, the device MUST execute the script, capture any output emitted from the script, and check if the script had any warnings or errors. If an error occurs while the device is connected to a trusted bootstrap server, the device MUST try to send a "post-script-error" progress report before exiting.

If the onboarding information was obtained from a trusted bootstrap server, and the result of the bootstrapping process did not disable the "flag to enable SZTP bootstrapping" described in Section 5.1, the device SHOULD send an "bootstrap-warning" progress report.

If the onboarding information was obtained from a trusted bootstrap server, the device MUST send a "bootstrap-complete" progress report. It is an error if the device does not receive back the "204 No Content" HTTP status line. If an error occurs, the device MUST try to send a "bootstrap-error" progress report before exiting.

At this point, the device has completely processed the bootstrapping data.

The device is now running its initial configuration. Notably, if NETCONF Call Home or RESTCONF Call Home [RFC8071] is configured, the

device initiates trying to establish the call home connections at this time.

Implementation Notes:

Implementations may vary in how to ensure no unwanted state is retained when an error occurs.

Following are some guidelines for if the implementation chooses to undo previous steps:

* When an error occurs, the device must rollback the current step and any previous steps.

* Most steps are atomic. For example, the processing of a configuration is specified above as atomic, and the processing of scripts is similarly specified as atomic in the "ietf-sztp-conveyed-info" YANG module.

* In case the error occurs after the initial configuration was committed, the device must restore the configuration to the configuration that existed prior to the configuration being committed.

* In case the error occurs after a script had executed successfully, it may be helpful for the implementation to define scripts as being able to take a conceptual input parameter indicating that the script should remove its previously set state.

6. The Conveyed Information Data Model

This section defines a YANG 1.1 [RFC7950] module that is used to define the data model for the conveyed information artifact described in Section 3.1. This data model uses the "yang-data" extension statement defined in [RFC8040]. Examples illustrating this data model are provided in Section 6.2.

6.1. Data Model Overview

The following tree diagram provides an overview of the data model for the conveyed information artifact.

```
module: ietf-sztp-conveyed-info

  yang-data conveyed-information:
    +-- (information-type)
       +--:(redirect-information)
       |  +-- redirect-information
       |     +-- bootstrap-server* [address]
       |        +-- address         inet:host
       |        +-- port?           inet:port-number
       |        +-- trust-anchor?   cms
       +--:(onboarding-information)
          +-- onboarding-information
             +-- boot-image
             |  +-- os-name?              string
             |  +-- os-version?           string
             |  +-- download-uri*         inet:uri
             |  +-- image-verification* [hash-algorithm]
             |     +-- hash-algorithm    identityref
             |     +-- hash-value        yang:hex-string
             +-- configuration-handling?      enumeration
             +-- pre-configuration-script?    script
             +-- configuration?               binary
             +-- post-configuration-script?   script
```

6.2. Example Usage

The following example illustrates how redirect information (Section 2.1) can be encoded using JSON.

```
{
  "ietf-sztp-conveyed-info:redirect-information" : {
    "bootstrap-server" : [
      {
        "address" : "sztp1.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      },
      {
        "address" : "sztp2.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      },
      {
        "address" : "sztp3.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      }
    ]
  }
}
```

The following example illustrates how onboarding information (Section 2.2) can be encoded using JSON.

[Note: '\' line wrapping for formatting only]

```
{
  "ietf-sztp-conveyed-info:onboarding-information" : {
    "boot-image" : {
      "os-name" : "VendorOS",
      "os-version" : "17.2R1.6",
      "download-uri" : [ "http://some/path/to/raw/file" ],
      "image-verification" : [
        {
          "hash-algorithm" : "ietf-sztp-conveyed-info:sha-256",
          "hash-value" : "ba:ec:cf:a5:67:82:b4:10:77:c6:67:a6:22:ab:\
7d:50:04:a7:8b:8f:0e:db:02:8b:f4:75:55:fb:c1:13:b2:33"
        }
      ]
    },
    "configuration-handling" : "merge",
    "pre-configuration-script" : "base64encodedvalue==",
    "configuration" : "base64encodedvalue==",
    "post-configuration-script" : "base64encodedvalue=="
  }
}
```

6.3. YANG Module

The conveyed information data model is defined by the YANG module presented in this section.

This module uses data types defined in [RFC5280], [RFC5652], [RFC6234], and [RFC6991], an extension statement from [RFC8040], and an encoding defined in [ITU.X690.2015].

```
<CODE BEGINS> file "ietf-sztp-conveyed-info@2019-01-15.yang"
module ietf-sztp-conveyed-info {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info";
  prefix sztp-info;

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-restconf {
    prefix rc;
    reference "RFC 8040: RESTCONF Protocol";
  }

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

  contact
    "WG Web:   http://tools.ietf.org/wg/netconf
     WG List:  <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kwatsen@juniper.net>";

  description
   "This module defines the data model for the Conveyed
    Information artifact defined in RFC XXXX: Secure Zero Touch
    Provisioning (SZTP).

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
    'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
    'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
    are to be interpreted as described in BCP 14 (RFC 2119,
    RFC 8174) when, and only when, they appear in all
    capitals, as shown here.
```

```
       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision 2019-01-15 {
       description
         "Initial version";
       reference
         "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
     }

     // identities

     identity hash-algorithm {
       description
         "A base identity for hash algorithm verification";
     }

     identity sha-256 {
       base "hash-algorithm";
       description "The SHA-256 algorithm.";
       reference "RFC 6234: US Secure Hash Algorithms.";
     }

     // typedefs

     typedef cms {
       type binary;
       description
         "A ContentInfo structure, as specified in RFC 5652,
          encoded using ASN.1 distinguished encoding rules (DER),
          as specified in ITU-T X.690.";
       reference
         "RFC 5652:
            Cryptographic Message Syntax (CMS)
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
```

```
     }

     // yang-data

     rc:yang-data "conveyed-information" {
       choice information-type {
         mandatory true;
         description
           "This choice statement ensures the response contains
            redirect-information or onboarding-information.";
         container redirect-information {
           description
             "Redirect information is described in Section 2.1 in
              RFC XXXX.  Its purpose is to redirect a device to
              another bootstrap server.";
           reference
             "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
           list bootstrap-server {
             key "address";
             min-elements 1;
             description
               "A bootstrap server entry.";
             leaf address {
               type inet:host;
               mandatory true;
               description
                "The IP address or hostname of the bootstrap server the
                 device should redirect to.";
             }
             leaf port {
               type inet:port-number;
               default "443";
               description
                "The port number the bootstrap server listens on.  If no
                 port is specified, the IANA-assigned port for 'https'
                 (443) is used.";
             }
             leaf trust-anchor {
               type cms;
               description
                 "A CMS structure that MUST contain the chain of
                  X.509 certificates needed to authenticate the TLS
                  certificate presented by this bootstrap server.

                  The CMS MUST only contain a single chain of
                  certificates.  The bootstrap server MUST only
                  authenticate to last intermediate CA certificate
                  listed in the chain.
```

```
                 In all cases, the chain MUST include a self-signed
                 root certificate.  In the case where the root
                 certificate is itself the issuer of the bootstrap
                 server's TLS certificate, only one certificate
                 is present.

                 If needed by the device, this CMS structure MAY
                 also contain suitably fresh revocation objects
                 with which the device can verify the revocation
                 status of the certificates.

                 This CMS encodes the degenerate form of the SignedData
                 structure that is commonly used to disseminate X.509
                 certificates and revocation objects (RFC 5280).";
               reference
                 "RFC 5280:
                    Internet X.509 Public Key Infrastructure Certificate
                    and Certificate Revocation List (CRL) Profile.";
             }
           }
         }
         container onboarding-information {
           description
             "Onboarding information is described in Section 2.2 in
              RFC XXXX.  Its purpose is to provide the device everything
              it needs to bootstrap itself.";
           reference
             "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
           container boot-image {
             description
               "Specifies criteria for the boot image the device MUST
                be running, as well as information enabling the device
                to install the required boot image.";
             leaf os-name {
               type string;
               description
                 "The name of the operating system software the device
                  MUST be running in order to not require a software
                  image upgrade (ex. VendorOS).";
             }
             leaf os-version {
               type string;
               description
                 "The version of the operating system software the
                  device MUST be running in order to not require a
                  software image upgrade (ex. 17.3R2.1).";
             }
             leaf-list download-uri {
```

```
              type inet:uri;
              ordered-by user;
              description
                "An ordered list of URIs to where the same boot image
                 file may be obtained.  How the URI schemes (http, ftp,
                 etc.) a device supports are known is vendor specific.
                 If a secure scheme (e.g., https) is provided, a device
                 MAY establish an untrusted connection to the remote
                 server, by blindly accepting the server's end-entity
                 certificate, to obtain the boot image.";
            }
            list image-verification {
              must '../download-uri' {
                description
                  "Download URIs must be provided if an image is to
                   be verified.";
              }
              key hash-algorithm;
              description
                "A list of hash values that a device can use to verify
                 boot image files with.";
              leaf hash-algorithm {
                type identityref {
                  base "hash-algorithm";
                }
                description
                  "Identifies the hash algorithm used.";
              }
              leaf hash-value {
                type yang:hex-string;
                mandatory true;
                description
                  "The hex-encoded value of the specified hash
                   algorithm over the contents of the boot image
                   file.";
              }
            }
          }
          leaf configuration-handling {
            type enumeration {
              enum "merge" {
                description
                  "Merge configuration into the running datastore.";
              }
              enum "replace" {
                description
                  "Replace the existing running datastore with the
                   passed configuration.";
```

```
              }
            }
            must '../configuration';
            description
              "This enumeration indicates how the server should process
               the provided configuration.";
          }
          leaf pre-configuration-script {
            type script;
            description
              "A script that, when present, is executed before the
               configuration has been processed.";
          }
          leaf configuration {
            type binary;
            must '../configuration-handling';
            description
              "Any configuration known to the device.  The use of
               the 'binary' type enables e.g., XML-content to be
               embedded into a JSON document.  The exact encoding
               of the content, as with the scripts, is vendor
               specific.";
          }
          leaf post-configuration-script {
            type script;
            description
              "A script that, when present, is executed after the
               configuration has been processed.";
          }
        }
      }
    }

    typedef script {
      type binary;
      description
        "A device specific script that enables the execution of
         commands to perform actions not possible thru configuration
         alone.

         No attempt is made to standardize the contents, running
         context, or programming language of the script, other than
         that it can indicate if any warnings or errors occurred and
         can emit output.  The contents of the script are considered
         specific to the vendor, product line, and/or model of the
         device.

         If the script execution indicates that an warning occurred,
```

```
          then the device MUST assume that the script had a soft error
          that the script believes will not affect manageability.

          If the script execution indicates that an error occurred,
          the device MUST assume the script had a hard error that the
          script believes will affect manageability.  In this case,
          the script is required to gracefully exit, removing any
          state that might hinder the device's ability to continue
          the bootstrapping sequence (e.g., process onboarding
          information obtained from another bootstrap server).";
     }
   }
   <CODE ENDS>
```

7. The SZTP Bootstrap Server API

This section defines the API for bootstrap servers. The API is defined as that produced by a RESTCONF [RFC8040] server that supports the YANG 1.1 [RFC7950] module defined in this section.

7.1. API Overview

The following tree diagram provides an overview for the bootstrap server RESTCONF API.

```
module: ietf-sztp-bootstrap-server

  rpcs:
    +---x get-bootstrapping-data
    |  +---w input
    |  |  +---w signed-data-preferred?   empty
    |  |  +---w hw-model?                string
    |  |  +---w os-name?                 string
    |  |  +---w os-version?              string
    |  |  +---w nonce?                   binary
    |  +--ro output
    |     +--ro reporting-level?    enumeration {onboarding-server}?
    |     +--ro conveyed-information    cms
    |     +--ro owner-certificate?      cms
    |     +--ro ownership-voucher?      cms
    +---x report-progress {onboarding-server}?
       +---w input
          +---w progress-type         enumeration
          +---w message?              string
          +---w ssh-host-keys
          |  +---w ssh-host-key* []
          |     +---w algorithm    string
          |     +---w key-data     binary
          +---w trust-anchor-certs
             +---w trust-anchor-cert*   cms
```

7.2. Example Usage

This section presents three examples illustrating the bootstrap server's API. Two examples are provided for the "get-bootstrapping-data" RPC (once to an untrusted bootstrap server, and again to a trusted bootstrap server), and one example for the "report-progress" RPC.

The following example illustrates a device using the API to fetch its bootstrapping data from a untrusted bootstrap server. In this example, the device sends the "signed-data-preferred" input parameter and receives signed data in the response.

REQUEST

[Note: '\' line wrapping for formatting only]

```
POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrappi\
ng-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <signed-data-preferred/>
</input>
```

RESPONSE

```
HTTP/1.1 200 OK
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+xml

<output
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <conveyed-information>base64encodedvalue==</conveyed-information>
  <owner-certificate>base64encodedvalue==</owner-certificate>
  <ownership-voucher>base64encodedvalue==</ownership-voucher>
</output>
```

The following example illustrates a device using the API to fetch its bootstrapping data from a trusted bootstrap server. In this example, the device sends addition input parameters to the bootstrap server, which it may use when formulating its response to the device.

REQUEST

[Note: '\' line wrapping for formatting only]

```
POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrappi\
ng-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <hw-model>model-x</hw-model>
  <os-name>vendor-os</os-name>
  <os-version>17.3R2.1</os-version>
  <nonce>extralongbase64encodedvalue=</nonce>
</input>
```

RESPONSE

```
HTTP/1.1 200 OK
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+xml

<output
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <reporting-level>verbose</reporting-level>
  <conveyed-information>base64encodedvalue==</conveyed-information>
</output>
```

The following example illustrates a device using the API to post a progress report to a bootstrap server. Illustrated below is the "bootstrap-complete" message, but the device may send other progress reports to the server while bootstrapping. In this example, the device is sending both its SSH host keys and a TLS server certificate, which the bootstrap server may, for example, pass to an NMS, as discussed in Appendix C.3.

REQUEST

[Note: '\' line wrapping for formatting only]

```
POST /restconf/operations/ietf-sztp-bootstrap-server:report-progress\
 HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
  <progress-type>bootstrap-complete</progress-type>
  <message>example message</message>
  <ssh-host-keys>
    <ssh-host-key>
      <algorithm>ssh-rsa</algorithm>
      <key-data>base64encodedvalue==</key-data>
    </ssh-host-key>
    <ssh-host-key>
      <algorithm>rsa-sha2-256</algorithm>
      <key-data>base64encodedvalue==</key-data>
    </ssh-host-key>
  </ssh-host-keys>
  <trust-anchor-certs>
    <trust-anchor-cert>base64encodedvalue==</trust-anchor-cert>
  </trust-anchor-certs>
</input>
```

RESPONSE

```
HTTP/1.1 204 No Content
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
```

7.3. YANG Module

The bootstrap server's device-facing API is normatively defined by the YANG module defined in this section.

This module uses data types defined in [RFC4253], [RFC5652], [RFC5280], [RFC6960], and [RFC8366], uses an encoding defined in [ITU.X690.2015], and makes a reference to [RFC4250] and [RFC6187].

```
<CODE BEGINS> file "ietf-sztp-bootstrap-server@2019-01-15.yang"
module ietf-sztp-bootstrap-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server";
  prefix sztp-svr;
```

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

     contact
       "WG Web:   <http://tools.ietf.org/wg/netconf/>
        WG List:  <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kwatsen@juniper.net>";

     description
      "This module defines an interface for bootstrap servers, as
       defined by RFC XXXX: Secure Zero Touch Provisioning (SZTP).

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
       'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
       'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
       are to be interpreted as described in BCP 14 (RFC 2119,
       RFC 8174) when, and only when, they appear in all
       capitals, as shown here.

       Copyright (c) 2019 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision 2019-01-15 {
       description
         "Initial version";
       reference
         "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
     }

     // features
     feature redirect-server {
       description
        "The server supports being a 'redirect server'.";
     }

     feature onboarding-server {
       description
        "The server supports being an 'onboarding server'.";
     }
```

```
     // typedefs

     typedef cms {
       type binary;
       description
         "A CMS structure, as specified in RFC 5652, encoded using
          ASN.1 distinguished encoding rules (DER), as specified in
          ITU-T X.690.";
       reference
         "RFC 5652:
            Cryptographic Message Syntax (CMS)
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
     }

     // RPCs

     rpc get-bootstrapping-data {
       description
         "This RPC enables a device, as identified by the RESTCONF
          username, to obtain bootstrapping data that has been made
          available for it.";
       input {
         leaf signed-data-preferred {
           type empty;
           description
             "This optional input parameter enables a device to
              communicate to the bootstrap server that it prefers
              to receive signed data.  Devices SHOULD always send
              this parameter when the bootstrap server is untrusted.
              Upon receiving this input parameter, the bootstrap
              server MUST return either signed data, or unsigned
              redirect information; the bootstrap server MUST NOT
              return unsigned onboarding information.";
         }
         leaf hw-model {
           type string;
           description
             "This optional input parameter enables a device to
              communicate to the bootstrap server its vendor specific
              hardware model number.  This parameter may be needed,
              for instance, when a device's IDevID certificate does
              not include the 'hardwareModelName' value in its
              subjectAltName field, as is allowed by 802.1AR-2009.";
           reference
```

```
            "IEEE 802.1AR-2009: IEEE Standard for Local and
               metropolitan area networks - Secure Device Identity";
        }
        leaf os-name {
          type string;
          description
            "This optional input parameter enables a device to
             communicate to the bootstrap server the name of its
             operating system.  This parameter may be useful if
             the device, as identified by its serial number, can
             run more than one type of operating system (e.g.,
             on a white-box system.";
        }
        leaf os-version {
          type string;
          description
            "This optional input parameter enables a device to
             communicate to the bootstrap server the version of its
             operating system.  This parameter may be used by a
             bootstrap server to return an operating system specific
             response to the device, thus negating the need for a
             potentially expensive boot-image update.";
        }
        leaf nonce {
          type binary {
            length "16..32";
          }
          description
            "This optional input parameter enables a device to
             communicate to the bootstrap server a nonce value.
             This may be especially useful for devices lacking
             an accurate clock, as then the bootstrap server
             can dynamically obtain from the manufacturer a
             voucher with the nonce value in it, as described
             in RFC 8366.";
          reference
            "RFC 8366:
               A Voucher Artifact for Bootstrapping Protocols";
        }
      }
      output {
        leaf reporting-level {
          if-feature onboarding-server;
          type enumeration {
            enum standard {
              description
                "Send just the progress reports required by RFC XXXX.";
              reference
```

```
                "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
            }
            enum verbose {
              description
                "Send additional progress reports that might help
                 troubleshooting an SZTP bootstrapping issue.";
            }
          }
          default standard;
          description
            "Specifies the reporting level for progress reports the
             bootstrap server would like to receive when processing
             onboarding information.  Progress reports are not sent
             when processing redirect information, or when the
             bootstrap server is untrusted (e.g., device sent the
             '<signed-data-preferred>' input parameter).";
        }
        leaf conveyed-information {
          type cms;
          mandatory true;
          description
            "An SZTP conveyed information artifact, as described in
             Section 3.1 of RFC XXXX.";
          reference
            "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
        }
        leaf owner-certificate {
          type cms;
          must '../ownership-voucher' {
            description
              "An ownership voucher must be present whenever an owner
               certificate is presented.";
          }
          description
            "An owner certificate artifact, as described in Section
             3.2 of RFC XXXX.  This leaf is optional because it is
             only needed when the conveyed information artifact is
             signed.";
          reference
            "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
        }
        leaf ownership-voucher {
          type cms;
          must '../owner-certificate' {
            description
              "An owner certificate must be present whenever an
               ownership voucher is presented.";
          }
```

```
          description
            "An ownership voucher artifact, as described by Section
             3.3 of RFC XXXX.  This leaf is optional because it is
             only needed when the conveyed information artifact is
             signed.";
          reference
            "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
        }
      }
    }

    rpc report-progress {
      if-feature onboarding-server;
      description
        "This RPC enables a device, as identified by the RESTCONF
         username, to report its bootstrapping progress to the
         bootstrap server.  This RPC is expected to be used when
         the device obtains onboarding-information from a trusted
         bootstap server.";
      input {
        leaf progress-type {
          type enumeration {
            enum "bootstrap-initiated" {
              description
                "Indicates that the device just used the
                 'get-bootstrapping-data' RPC.  The 'message' node
                 below MAY contain any additional information that
                 the manufacturer thinks might be useful.";
            }
            enum "parsing-initiated" {
              description
                "Indicates that the device is about to start parsing
                 the onboarding information.  This progress type is
                 only for when parsing is implemented as a distinct
                 step.";
            }
            enum "parsing-warning" {
              description
                "Indicates that the device had a non-fatal error when
                 parsing the response from the bootstrap server.  The
                 'message' node below SHOULD indicate the specific
                 warning that occurred.";
            }
            enum "parsing-error" {
              description
                "Indicates that the device encountered a fatal error
                 when parsing the response from the bootstrap server.
                 For instance, this could be due to malformed encoding,
```

```
                 the device expecting signed data when only unsigned
                 data is provided, the ownership voucher not listing
                 the device's serial number, or because the signature
                 didn't match.  The 'message' node below SHOULD
                 indicate the specific error.  This progress type
                 also indicates that the device has abandoned trying
                 to bootstrap off this bootstrap server.";
            }
            enum "parsing-complete" {
              description
                "Indicates that the device successfully completed
                 parsing the onboarding information.  This progress
                 type is only for when parsing is implemented as a
                 distinct step.";
            }
            enum "boot-image-initiated" {
              description
                "Indicates that the device is about to start
                 processing the boot-image information.";
            }
            enum "boot-image-warning" {
              description
                "Indicates that the device encountered a non-fatal
                 error condition when trying to install a boot-image.
                 A possible reason might include a need to reformat a
                 partition causing loss of data.  The 'message' node
                 below SHOULD indicate any warning messages that were
                 generated.";
            }
            enum "boot-image-error" {
              description
                "Indicates that the device encountered an error when
                 trying to install a boot-image, which could be for
                 reasons such as a file server being unreachable,
                 file not found, signature mismatch, etc.  The
                 'message' node SHOULD indicate the specific error
                 that occurred.  This progress type also indicates
                 that the device has abandoned trying to bootstrap
                 off this bootstrap server.";
            }
            enum "boot-image-mismatch" {
              description
                "Indicates that the device that has determined that
                 it is not running the correct boot image.  This
                 message SHOULD precipitate trying to download
                 a boot image.";
            }
            enum "boot-image-installed-rebooting" {
```

```
                description
                  "Indicates that the device successfully installed
                   a new boot image and is about to reboot.  After
                   sending this progress type, the device is not
                   expected to access the bootstrap server again
                   for this bootstrapping attempt.";
              }
              enum "boot-image-complete" {
                description
                  "Indicates that the device believes that it is
                   running the correct boot-image.";
              }
              enum "pre-script-initiated" {
                description
                  "Indicates that the device is about to execute the
                   'pre-configuration-script'.";
              }
              enum "pre-script-warning" {
                description
                  "Indicates that the device obtained a warning from the
                   'pre-configuration-script' when it was executed.  The
                   'message' node below SHOULD capture any output the
                   script produces.";
              }
              enum "pre-script-error" {
                description
                  "Indicates that the device obtained an error from the
                   'pre-configuration-script' when it was executed. The
                   'message' node below SHOULD capture any output the
                   script produces.  This progress type also indicates
                   that the device has abandoned trying to bootstrap
                   off this bootstrap server.";
              }
              enum "pre-script-complete" {
                description
                  "Indicates that the device successfully executed the
                   'pre-configuration-script'.";
              }
              enum "config-initiated" {
                description
                  "Indicates that the device is about to commit the
                   initial configuration.";
              }
              enum "config-warning" {
                description
                  "Indicates that the device obtained warning messages
                   when it committed the initial configuration.  The
                   'message' node below SHOULD indicate any warning
```

```
                 messages that were generated.";
             }
             enum "config-error" {
               description
                 "Indicates that the device obtained error messages
                  when it committed the initial configuration.  The
                  'message' node below SHOULD indicate the error
                  messages that were generated.  This progress type
                  also indicates that the device has abandoned trying
                  to bootstrap off this bootstrap server.";
             }
             enum "config-complete" {
               description
                 "Indicates that the device successfully committed
                  the initial configuration.";
             }
             enum "post-script-initiated" {
               description
                 "Indicates that the device is about to execute the
                  'post-configuration-script'.";
             }
             enum "post-script-warning" {
               description
                 "Indicates that the device obtained a warning from the
                  'post-configuration-script' when it was executed. The
                  'message' node below SHOULD capture any output the
                  script produces.";
             }
             enum "post-script-error" {
               description
                 "Indicates that the device obtained an error from the
                  'post-configuration-script' when it was executed. The
                  'message' node below SHOULD capture any output the
                  script produces.  This progress type also indicates
                  that the device has abandoned trying to bootstrap
                  off this bootstrap server.";
             }
             enum "post-script-complete" {
               description
                 "Indicates that the device successfully executed the
                  'post-configuration-script'.";
             }
             enum "bootstrap-warning" {
              description
                "Indicates that a warning condition occurred for which
                 there no other 'progress-type' enumeration is deemed
                 suitable.  The 'message' node below SHOULD describe
                 the warning.";
```

```
            }
            enum "bootstrap-error" {
             description
               "Indicates that an error condition occurred for which
                there no other 'progress-type' enumeration is deemed
                suitable.  The 'message' node below SHOULD describe
                the error.  This progress type also indicates that
                the device has abandoned trying to bootstrap off
                this bootstrap server.";
            }
            enum "bootstrap-complete" {
              description
                "Indicates that the device successfully processed
                 all 'onboarding-information' provided, and that it
                 is ready to be managed.  The 'message' node below
                 MAY contain any additional information that the
                 manufacturer thinks might be useful.  After sending
                 this progress type, the device is not expected to
                 access the bootstrap server again.";
            }
            enum "informational" {
              description
                "Indicates any additional information not captured
                 by any of the other progress types. For instance,
                 a message indicating that the device is about to
                 reboot after having installed a boot-image could
                 be provided.  The 'message' node below SHOULD
                 contain information that the manufacturer thinks
                 might be useful.";
            }
          }
          mandatory true;
          description
            "The type of progress report provided.";
        }
        leaf message {
          type string;
          description
            "An optional arbitrary value.";
        }
        container ssh-host-keys {
          when "../progress-type = 'bootstrap-complete'" {
            description
              "SSH host keys are only sent when the progress type
               is 'bootstrap-complete'.";
          }
          description
            "A list of SSH host keys an NMS may use to authenticate
```

```
             subsequent SSH-based connections to this device (e.g.,
             netconf-ssh, netconf-ch-ssh).";
          list ssh-host-key {
            description
              "An SSH host key an NMS may use to authenticate
               subsequent SSH-based connections to this device
               (e.g., netconf-ssh, netconf-ch-ssh).";
            reference
              "RFC 4253: The Secure Shell (SSH) Transport Layer
                         Protocol";
            leaf algorithm {
              type string;
              mandatory true;
              description
                "The public key algorithm name for this SSH key.

                 Valid values are listed in the 'Public Key Algorithm
                 Names' subregistry of the 'Secure Shell (SSH) Protocol
                 Parameters' registry maintained by IANA.";
              reference
                "RFC 4250: The Secure Shell (SSH) Protocol Assigned
                           Numbers
                 IANA URL: https://www.iana.org/assignments/ssh-param\\
                           eters/ssh-parameters.xhtml#ssh-parameters-19
                           ('\\' added for formatting reasons)";
            }
            leaf key-data {
              type binary;
              mandatory true;
              description
                "The binary public key data for this SSH key, as
                 specified by RFC 4253, Section 6.6, i.e.:

                   string    certificate or public key format
                             identifier
                   byte[n]   key/certificate data.";
              reference
                "RFC 4253: The Secure Shell (SSH) Transport Layer
                           Protocol";
            }
          }
        }
        container trust-anchor-certs {
          when "../progress-type = 'bootstrap-complete'" {
            description
              "Trust anchors are only sent when the progress type
               is 'bootstrap-complete'.";
          }
```

```
          description
            "A list of trust anchor certificates an NMS may use to
             authenticate subsequent certificate-based connections
             to this device (e.g., restconf-tls, netconf-tls, or
             even netconf-ssh with X.509 support from RFC 6187).
             In practice, trust anchors for IDevID certificates do
             not need to be conveyed using this mechanism.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication.";
          leaf-list trust-anchor-cert {
            type cms;
            description
             "A CMS structure whose top-most content type MUST be the
              signed-data content type, as described by Section 5 in
              RFC 5652.

              The CMS MUST contain the chain of X.509 certificates
              needed to authenticate the certificate presented by
              the device.

              The CMS MUST contain only a single chain of
              certificates.  The last certificate in the chain
              MUST be the issuer for the device's end-entity
              certificate.

              In all cases, the chain MUST include a self-signed
              root certificate.  In the case where the root
              certificate is itself the issuer of the device's
              end-entity certificate, only one certificate is
              present.

              This CMS encodes the degenerate form of the SignedData
              structure that is commonly used to disseminate X.509
              certificates and revocation objects (RFC 5280).";
            reference
              "RFC 5280:
                 Internet X.509 Public Key Infrastructure
                 Certificate and Certificate Revocation List (CRL)
                 Profile.
               RFC 5652:
                 Cryptographic Message Syntax (CMS)";
          }
        }
      }
    }
  }
  <CODE ENDS>
```

8. DHCP Options

This section defines two DHCP options, one for DHCPv4 and one for DHCPv6. These two options are semantically the same, though syntactically different.

8.1. DHCPv4 SZTP Redirect Option

The DHCPv4 SZTP Redirect Option is used to provision the client with one or more URIs for bootstrap servers that can be contacted to attempt further configuration.

```
DHCPv4 SZTP Redirect Option

 0                              1
 0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|   option-code (143)   |     option-length     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
.                                               .
.    bootstrap-server-list (variable length)    .
.                                               .
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

* option-code: OPTION_V4_SZTP_REDIRECT (143)
* option-length: The option length in octets.
* bootstrap-server-list: A list of servers for the
   client to attempt contacting, in order to obtain
   further bootstrapping data, in the format shown
   in Section 8.3.
```

DHCPv4 Client Behavior

Clients MAY request the OPTION_V4_SZTP_REDIRECT by including its option code in the Parameter Request List (55) in DHCP request messages.

On receipt of a DHCPv4 Reply message which contains the OPTION_V4_SZTP_REDIRECT, the client processes the response according to Section 5.5, with the understanding that the "address" and "port" values are encoded in the URIs.

Any invalid URI entries received in the uri-data field are ignored by the client. If OPTION_V4_SZTP_REDIRECT does not contain at least one valid URI entry in the uri-data field, then the client MUST discard the option.

As the list of URIs may exceed the maximum allowed length of a single DHCPv4 option (255 octets), the client MUST implement [RFC3396], allowing the URI list to be split across a number of OPTION_V4_SZTP_REDIRECT option instances.

DHCPv4 Server Behavior

The DHCPv4 server MAY include a single instance of Option OPTION_V4_SZTP_REDIRECT in DHCP messages it sends. Servers MUST NOT send more than one instance of the OPTION_V4_SZTP_REDIRECT option.

The server's DHCP message MUST contain only a single instance of the OPTION_V4_SZTP_REDIRECT's 'bootstrap-server-list' field. However, the list of URIs in this field may exceed the maximum allowed length of a single DHCPv4 option (per [RFC3396]).

If the length of 'bootstrap-server-list' is small enough to fit into a single instance of OPTION_V4_SZTP_REDIRECT, the server MUST NOT send more than one instance of this option.

If the length of the 'bootstrap-server-list' field is too large to fit into a single option, then OPTION_V4_SZTP_REDIRECT MUST be split into multiple instances of the option according to the process described in [RFC3396].

8.2. DHCPv6 SZTP Redirect Option

The DHCPv6 SZTP Redirect Option is used to provision the client with one or more URIs for bootstrap servers that can be contacted to attempt further configuration.

DHCPv6 SZTP Redirect Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       option-code (136)       |          option-length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.           bootstrap-server-list (variable length)             .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* option-code: OPTION_V6_SZTP_REDIRECT (136)
* option-length: The option length in octets.
* bootstrap-server-list: A list of servers for the client to attempt contacting, in order to obtain further bootstrapping data, in the format shown in Section 8.3.

DHCPv6 Client Behavior

Clients MAY request the OPTION_V6_SZTP_REDIRECT option, as defined in [RFC8415], Sections 18.2.1, 18.2.2, 18.2.4, 18.2.5, 18.2.6, and 21.7. As a convenience to the reader, we mention here that the client includes requested option codes in the Option Request Option.

On receipt of a DHCPv6 Reply message which contains the OPTION_V6_SZTP_REDIRECT, the client processes the response according to Section 5.5, with the understanding that the "address" and "port" values are encoded in the URIs.

Any invalid URI entries received in the uri-data field are ignored by the client. If OPTION_V6_SZTP_REDIRECT does not contain at least one valid URI entry in the uri-data field, then the client MUST discard the option.

DHCPv6 Server Behavior

Section 18.3 of [RFC8415] governs server operation in regard to option assignment. As a convenience to the reader, we mention here that the server will send a particular option code only if configured with specific values for that option code and if the client requested it.

Option OPTION_V6_SZTP_REDIRECT is a singleton. Servers MUST NOT send more than one instance of the OPTION_V6_SZTP_REDIRECT option.

8.3. Common Field Encoding

Both of the DHCPv4 and DHCPv6 options defined in this section encode a list of bootstrap server URIs. The "URI" structure is a DHCP option that can contain multiple URIs (see [RFC7227], Section 5.7). Each URI entry in the bootstrap-server-list is structured as follows:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...-+-+-+-+-+-+-+
 |       uri-length              |          URI                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...-+-+-+-+-+-+-+
```

* uri-length: 2 octets long, specifies the length of the URI data.
* URI: URI of SZTP bootstrap server.

The URI of the SZTP bootstrap server MUST use the "https" URI scheme defined in Section 2.7.2 of [RFC7230], and MUST be in form "https://<ip-address-or-hostname>[:<port>]".

9. Security Considerations

9.1. Clock Sensitivity

The solution in this document relies on TLS certificates, owner certificates, and ownership vouchers, all of which require an accurate clock in order to be processed correctly (e.g., to test validity dates and revocation status). Implementations SHOULD ensure devices have an accurate clock when shipped from manufacturing facilities, and take steps to prevent clock tampering.

If it is not possible to ensure clock accuracy, it is RECOMMENDED that implementations disable the aspects of the solution having clock sensitivity. In particular, such implementations should assume that TLS certificates, ownership vouchers, and owner certificates never expire and are not revokable. From an ownership voucher perspective, manufacturers SHOULD issue a single ownership voucher for the lifetime of such devices.

Implementations SHOULD NOT rely on NTP for time, as NTP is not a secure protocol at this time. Note, there is an IETF work-in-progress to secure NTP [I-D.ietf-ntp-using-nts-for-ntp].

9.2. Use of IDevID Certificates

IDevID certificates, as defined in [Std-802.1AR-2018], are RECOMMENDED, both for the TLS-level client certificate used by devices when connecting to a bootstrap server, as well as for the device identity certificate used by owners when encrypting the SZTP bootstrapping data artifacts.

9.3. Immutable Storage for Trust Anchors

Devices MUST ensure that all their trust anchor certificates, including those for connecting to bootstrap servers and verifying ownership vouchers, are protected from external modification.

It may be necessary to update these certificates over time (e.g., the manufacturer wants to delegate trust to a new CA). It is therefore expected that devices MAY update these trust anchors when needed through a verifiable process, such as a software upgrade using signed software images.

9.4. Secure Storage for Long-lived Private Keys

Manufacturer-generated device identifiers may have very long lifetimes. For instance, [Std-802.1AR-2018] recommends using the "notAfter" value 99991231235959Z in IDevID certificates. Given the

long-lived nature of these private keys, it is paramount that they are stored so as to resist discovery, such as in a secure cryptographic processor, such as a trusted platform module (TPM) chip.

9.5. Blindly Authenticating a Bootstrap Server

This document allows a device to blindly authenticate a bootstrap server's TLS certificate. It does so to allow for cases where the redirect information may be obtained in an unsecured manner, which is desirable to support in some cases.

To compensate for this, this document requires that devices, when connected to an untrusted bootstrap server, assert that data downloaded from the server is signed.

9.6. Disclosing Information to Untrusted Servers

This document allows devices to establish connections to untrusted bootstrap servers. However, since the bootstrap server is untrusted, it may be under the control of an adversary, and therefore devices SHOULD be cautious about the data they send to the bootstrap server in such cases.

Devices send different data to bootstrap servers at each of the protocol layers TCP, TLS, HTTP, and RESTCONF.

At the TCP protocol layer, devices may relay their IP address, subject to network translations. Disclosure of this information is not considered a security risk.

At the TLS protocol layer, devices may use a client certificate to identify and authenticate themselves to untrusted bootstrap servers. At a minimum, the client certificate must disclose the device's serial number, and may disclose additional information such as the device's manufacturer, hardware model, public key, etc. Knowledge of this information may provide an adversary with details needed to launch an attack. It is RECOMMENDED that secrecy of the network constituency is not relied on for security.

At the HTTP protocol layer, devices may use an HTTP authentication scheme to identify and authenticate themselves to untrusted bootstrap servers. At a minimum, the authentication scheme must disclose the device's serial number and, concerningly, may, depending on the authentication mechanism used, reveal a secret that is only supposed to be known to the device (e.g., a password). Devices SHOULD NOT use an HTTP authentication scheme (e.g., HTTP Basic) with an untrusted

bootstrap server that reveals a secret that is only supposed to be known to the device.

At the RESTCONF protocol layer, devices use the "get-bootstrapping-data" RPC, but not the "report-progress" RPC, when connected to an untrusted bootstrap server. The "get-bootstrapping-data" RPC allows additional input parameters to be passed to the bootstrap server (e.g., "os-name", "os-version", "hw-model"). It is RECOMMENDED that devices only pass the "signed-data-preferred" input parameter to an untrusted bootstrap server. While it is okay for a bootstrap server to immediately return signed onboarding information, it is RECOMMENDED that bootstrap servers instead promote the untrusted connection to a trusted connection, as described in Appendix B, thus enabling the device to use the "report-progress" RPC while processing the onboarding information.

9.7. Sequencing Sources of Bootstrapping Data

For devices supporting more than one source for bootstrapping data, no particular sequencing order has to be observed for security reasons, as the solution for each source is considered equally secure. However, from a privacy perspective, it is RECOMMENDED that devices access local sources before accessing remote sources.

9.8. Safety of Private Keys used for Trust

The solution presented in this document enables bootstrapping data to be trusted in two ways, either through transport level security or through the signing of artifacts.

When transport level security (i.e., a trusted bootstrap server) is used, the private key for the end-entity certificate must be online in order to establish the TLS connection.

When artifacts are signed, the signing key is required to be online only when the bootstrap server is returning a dynamically generated signed-data response. For instance, a bootstrap server, upon receiving the "signed-data-preferred" input parameter to the "get-bootstrapping-data" RPC, may dynamically generate a response that is signed.

Bootstrap server administrators are RECOMMENDED to follow best practice to protect the private key used for any online operation. For instance, use of a hardware security module (HSM) is RECOMMENDED. If an HSM is not used, frequent private key refreshes are RECOMMENDED, assuming all bootstrapping devices have an accurate clock (see Section 9.1).

For best security, it is RECOMMENDED that owners only provide bootstrapping data that has been signed, using a protected private key, and encrypted, using the device's public key from its secure device identity certificate.

9.9. Increased Reliance on Manufacturers

The SZTP bootstrapping protocol presented in this document shifts some control of initial configuration away from the rightful owner of the device and towards the manufacturer and its delegates.

The manufacturer maintains the list of well-known bootstrap servers its devices will trust. By design, if no bootstrapping data is found via other methods first, the device will try to reach out to the well-known bootstrap servers. There is no mechanism to prevent this from occurring other than by using an external firewall to block such connections. Concerns related to trusted bootstrap servers are discussed in Section 9.10.

Similarly, the manufacturer maintains the list of voucher signing authorities its devices will trust. The voucher signing authorities issue the vouchers that enable a device to trust an owner's domain certificate. It is vital that manufacturers ensure the integrity of these voucher signing authorities, so as to avoid incorrect assignments.

Operators should be aware that this system assumes that they trust all the pre-configured bootstrap servers and voucher signing authorities designated by the manufacturers. While operators may use points in the network to block access to the well-known bootstrap servers, operators cannot prevent voucher signing authorities from generating vouchers for their devices.

9.10. Concerns with Trusted Bootstrap Servers

Trusted bootstrap servers, whether well-known or discovered, have the potential to cause problems, such as the following.

- o A trusted bootstrap server that has been compromised may be modified to return unsigned data of any sort. For instance, a bootstrap server that is only suppose to return redirect information might be modified to return onboarding information. Similarly, a bootstrap server that is only supposed to return signed data, may be modified to return unsigned data. In both cases, the device will accept the response, unaware that it wasn't supposed to be any different. It is RECOMMENDED that maintainers of trusted bootstrap servers ensure that their systems are not easily compromised and, in case of compromise, have mechanisms in

place to detect and remediate the compromise as expediently as possible.

- o A trusted bootstrap server hosting either unsigned, or signed but not encrypted, data may disclose information to unwanted parties (e.g., an administrator of the bootstrap server). This is a privacy issue only, but could reveal information that might be used in a subsequent attack. Disclosure of redirect information has limited exposure (it is just a list of bootstrap servers), whereas disclosure of onboarding information could be highly revealing (e.g., network topology, firewall policies, etc.). It is RECOMMENDED that operators encrypt the bootstrapping data when its contents are considered sensitive, even to the point of hiding it from the administrators of the bootstrap server, which may be maintained by a 3rd-party.

9.11. Validity Period for Conveyed Information

The conveyed information artifact does not specify a validity period. For instance, neither redirect information nor onboarding information enable "not-before" or "not-after" values to be specified, and neither artifact alone can be revoked.

For unsigned data provided by an untrusted source of bootstrapping data, it is not meaningful to discuss its validity period when the information itself has no authenticity and may have come from anywhere.

For unsigned data provided by a trusted source of bootstrapping data (i.e., a bootstrap server), the availability of the data is the only measure of it being current. Since the untrusted data comes from a trusted source, its current availability is meaningful and, since bootstrap servers use TLS, the contents of the exchange cannot be modified or replayed.

For signed data, whether provided by an untrusted or trusted source of bootstrapping data, the validity is constrained by the validity of the both the ownership voucher and owner certificate used to authenticate it.

The ownership voucher’s validity is primarily constrained by the ownership voucher’s "created-on" and "expires-on" nodes. While [RFC8366] recommends short-lived vouchers (see Section 6.1), the "expires-on" node may be set to any point in the future, or omitted altogether to indicate that the voucher never expires. The ownership voucher’s validity is secondarily constrained by the manufacturer’s PKI used to sign the voucher; whilst an ownership voucher cannot be revoked directly, the PKI used to sign it may be.

The owner certificate's validity is primarily constrained by the X.509's validity field, the "notBefore" and "notAfter" values, as specified by the certificate authority that signed it. The owner certificate's validity is secondarily constrained by the validity of the PKI used to sign the voucher. Owner certificates may be revoked directly.

For owners that wish to have maximum flexibility in their ability to specify and constrain the validity of signed data, it is RECOMMENDED that a unique owner certificate is created for each signed artifact. Not only does this enable a validity period to be specified, for each artifact, but it also enables to the validity of each artifact to be revoked.

9.12. Cascading Trust via Redirects

Redirect Information (Section 2.1), by design, instructs a bootstrapping device to initiate a HTTPS connection to the specified bootstrap servers.

When the redirect information is trusted, the redirect information can encode a trust anchor certificate used by the device to authenticate the TLS end-entity certificate presented by each bootstrap server.

As a result, any compromise in an interaction providing redirect information may result in compromise of all subsequent interactions.

9.13. Possible Reuse of Private Keys

This document describes two uses for secure device identity certificates.

The primary use is for when the device authenticates itself to a bootstrap server, using its private key for TLS-level client-certificate based authentication.

A secondary use is for when the device needs to decrypt provided bootstrapping artifacts, using its private key to decrypt the data or, more precisely, per Section 6 in [RFC5652], decrypt a symmetric key used to decrypt the data.

This document, in Section 3.4 allows for the possibility that the same secure device identity certificate is used for both uses, as [Std-802.1AR-2018] states that a DevID certificate MAY have the "keyEncipherment" KeyUsage bit, in addition to the "digitalSignature" KeyUsage bit, set.

While it is understood that it is generally frowned upon to reuse private keys, this document views such reuse acceptable as there are not any known ways to cause a signature made in one context to be (mis)interpreted as valid in the other context.

9.14. Non-Issue with Encrypting Signed Artifacts

This document specifies the encryption of signed objects, as opposed to the signing of encrypted objects, as might be expected given well-publicized oracle attacks (e.g., the padding oracle attack).

This document does not view such attacks as feasible in the context of the solution because the decrypted text never leaves the device.

9.15. The "ietf-sztp-conveyed-info" YANG Module

The ietf-sztp-conveyed-info module defined in this document defines a data structure that is always wrapped by a CMS structure. When accessed by a secure mechanism (e.g., protected by TLS), then the CMS structure may be unsigned. However, when accessed by an insecure mechanism (e.g., removable storage device), then the CMS structure must be signed, in order for the device to trust it.

Implementations should be aware that signed bootstrapping data only protects the data from modification, and that the contents are still visible to others. This doesn’t affect security so much as privacy. That the contents may be read by unintended parties when accessed by insecure mechanisms is considered next.

The ietf-sztp-conveyed-info module defines a top-level "choice" statement that declares the contents are either "redirect-information" or "onboarding-information". Each of these two cases are now considered.

When the content of the CMS structure is redirect-information, an observer can learn about the bootstrap servers the device is being directed to, their IP addresses or hostnames, ports, and trust anchor certificates. Knowledge of this information could provide an observer some insight into a network’s inner structure.

When the content of the CMS structure is onboarding information, an observer could learn considerable information about how the device is to be provisioned. This information includes the operating system version, initial configuration, and script contents. This information should be considered sensitive and precautions should be taken to protect it (e.g., encrypt the artifact using the device’s public key).

9.16. The "ietf-sztp-bootstrap-server" YANG Module

The ietf-sztp-bootstrap-server module defined in this document specifies an API for a RESTCONF [RFC8040]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a preconfigured subset of all available protocol operations and content.

This module presents no data nodes (only RPCs). There is no need to discuss the sensitivity of data nodes.

This module defines two RPC operations that may be considered sensitive in some network environments. These are the operations and their sensitivity/vulnerability:

get-bootstrapping-data: This RPC is used by devices to obtain their bootstrapping data. By design, each device, as identified by its authentication credentials (e.g. client certificate), can only obtain its own data. NACM is not needed to further constrain access to this RPC.

report-progress: This RPC is used by devices to report their bootstrapping progress. By design, each device, as identified by its authentication credentials (e.g. client certificate), can only report data for itself. NACM is not needed to further constrain access to this RPC.

10. IANA Considerations

10.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688] maintained at https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

10.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020] maintained at https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml. Following the format defined in [RFC6020], the below registrations are requested:

```
name:      ietf-sztp-conveyed-info
namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info
prefix:    sztp-info
reference: RFC XXXX

name:      ietf-sztp-bootstrap-server
namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server
prefix:    sztp-svr
reference: RFC XXXX
```

10.3. The SMI Security for S/MIME CMS Content Type Registry

This document registers two SMI security codes in the "SMI Security for S/MIME CMS Content Type" registry (1.2.840.113549.1.9.16.1) maintained at https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#security-smime-1. Following the format used in Section 3.4 of [RFC7107], the below registrations are requested:

Decimal	Description	References
TBD1	id-ct-sztpConveyedInfoXML	[RFCXXXX]
TBD2	id-ct-sztpConveyedInfoJSON	[RFCXXXX]

id-ct-sztpConveyedInfoXML indicates that the "conveyed-information" is encoded using XML. id-ct-sztpConveyedInfoJSON indicates that the "conveyed-information" is encoded using JSON.

10.4. The BOOTP Manufacturer Extensions and DHCP Options Registry

This document registers one DHCP code point in the "BOOTP Manufacturer Extensions and DHCP Options" registry maintained at http://www.iana.org/assignments/bootp-dhcp-parameters. Following the format used by other registrations, the below registration is requested:

```
Tag:         143
Name:        OPTION_V4_SZTP_REDIRECT
Data Length: N
Meaning:     This option provides a list of URIs
             for SZTP bootstrap servers
Reference:   [RFCXXXX]
```

Note: this request is to make permanent a previously registered early code point allocation.

10.5. The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Registry

This document registers one DHCP code point in "Option Codes" subregistry of the "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" registry maintained at http://www.iana.org/assignments/dhcpv6-parameters. Following the format used by other registrations, the below registration is requested:

```
Value:            136
Description:      OPTION_V6_SZTP_REDIRECT
Client ORO:       Yes
Singleton Option: Yes
Reference:        [RFCXXXX]
```

Note: this request is to make permanent a previously registered early code point allocation.

10.6. The Service Name and Transport Protocol Port Number Registry

This document registers one service name in the Service Name and Transport Protocol Port Number Registry [RFC6335] maintained at https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. Following the format defined in Section 8.1.1 of [RFC6335], the below registration is requested:

```
Service Name:            sztp
Transport Protocol(s):   TCP
Assignee:                IESG <iesg@ietf.org>
Contact:                 IETF Chair <chair@ietf.org>
Description:             This service name is used to construct the
                         SRV service label "_sztp" for discovering
                         SZTP bootstrap servers.
Reference:               [RFCXXXX]
Port Number:             N/A
Service Code:            N/A
Known Unauthorized Uses: N/A
Assignment Notes:        This protocol uses HTTPS as a substrate.
```

10.7. The DNS Underscore Global Scoped Entry Registry

This document registers one service name in the DNS Underscore Global Scoped Entry Registry [I-D.ietf-dnsop-attrleaf] maintained at TBD_IANA_URL. Following the format defined in Section 4.3 of [I-D.ietf-dnsop-attrleaf], the below registration is requested:

```
RR Type:            TXT
_NODE NAME:         _sztp
Reference:          [RFCXXXX]
```

11. References

11.1. Normative References

[I-D.ietf-dnsop-attrleaf]
Crocker, D., "DNS Scoped Data Through "Underscore" Naming of Attribute Leaves", draft-ietf-dnsop-attrleaf-16 (work in progress), November 2018.

[ITU.X690.2015]
International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1, August 2015, <https://www.itu.int/rec/T-REC-X.690/>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4)", RFC 3396, DOI 10.17487/RFC3396, November 2002, <https://www.rfc-editor.org/info/rfc3396>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <https://www.rfc-editor.org/info/rfc7227>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/info/rfc8366>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

[Std-802.1AR-2018] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", June 2018, <https://standards.ieee.org/standard/802_1AR-2018.html>.

11.2. Informative References

[I-D.ietf-netconf-crypto-types] Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-02 (work in progress), October 2018.

[I-D.ietf-netconf-trust-anchors] Watsen, K., "YANG Data Model for Global Trust Anchors", draft-ietf-netconf-trust-anchors-02 (work in progress), October 2018.

[I-D.ietf-ntp-using-nts-for-ntp] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", draft-ietf-ntp-using-nts-for-ntp-15 (work in progress), December 2018.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, DOI 10.17487/RFC4250, January 2006, <https://www.rfc-editor.org/info/rfc4250>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/info/rfc6891>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC7107] Housley, R., "Object Identifier Registry for the S/MIME Mail Security Working Group", RFC 7107, DOI 10.17487/RFC7107, January 2014, <https://www.rfc-editor.org/info/rfc7107>.

[RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <https://www.rfc-editor.org/info/rfc7766>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Example Device Data Model

This section defines a non-normative data model that enables the configuration of SZTP bootstrapping and discovery of what parameters are used by a device's bootstrapping logic.

A.1. Data Model Overview

The following tree diagram provides an overview for the SZTP device data model.

```
 module: example-device-data-model
   +--rw sztp
      +--rw enabled?                          boolean
      +--ro idevid-certificate?               ct:end-entity-cert-cms
      |       {bootstrap-servers}?
      +--ro bootstrap-servers {bootstrap-servers}?
      |  +--ro bootstrap-server* [address]
      |     +--ro address    inet:host
      |     +--ro port?      inet:port-number
      +--ro bootstrap-server-trust-anchors {bootstrap-servers}?
      |  +--ro reference*   ta:pinned-certificates-ref
      +--ro voucher-trust-anchors {signed-data}?
         +--ro reference*   ta:pinned-certificates-ref
```

In the above diagram, notice that there is only one configurable node "enabled". The expectation is that this node would be set to "true" in device's factory default configuration and that it would either be set to "false" or deleted when the SZTP bootstrapping is longer needed.

A.2. Example Usage

Following is an instance example for this data model.

```
<sztp xmlns="https://example.com/sztp-device-data-model">
  <enabled>true</enabled>
  <idevid-certificate>base64encodedvalue==</idevid-certificate>
  <bootstrap-servers>
    <bootstrap-server>
      <address>sztp1.example.com</address>
      <port>8443</port>
    </bootstrap-server>
    <bootstrap-server>
      <address>sztp2.example.com</address>
      <port>8443</port>
    </bootstrap-server>
    <bootstrap-server>
      <address>sztp3.example.com</address>
      <port>8443</port>
    </bootstrap-server>
  </bootstrap-servers>
  <bootstrap-server-trust-anchors>
    <reference>manufacturers-root-ca-certs</reference>
  </bootstrap-server-trust-anchors>
  <voucher-trust-anchors>
    <reference>manufacturers-root-ca-certs</reference>
  </voucher-trust-anchors>
</sztp>
```

A.3. YANG Module

The device model is defined by the YANG module defined in this section.

This module uses data types defined in [RFC6991], [I-D.ietf-netconf-crypto-types], and [I-D.ietf-netconf-trust-anchors].

```
module example-device-data-model {
  yang-version 1.1;
  namespace "https://example.com/sztp-device-data-model";
  prefix sztp-ddm;

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-crypto-types {
    prefix ct;
    revision-date 2018-06-04;
    description
```

```
        "This revision is defined in the -00 version of
         draft-ietf-netconf-crypto-types";
      reference
       "draft-ietf-netconf-crypto-types:
          Common YANG Data Types for Cryptography";
    }

    import ietf-trust-anchors {
      prefix ta;
      revision-date 2018-06-04;
      description
       "This revision is defined in -00 version of
        draft-ietf-netconf-trust-anchors.";
      reference
       "draft-ietf-netconf-trust-anchors:
          YANG Data Model for Global Trust Anchors";
    }

    organization
      "Example Corporation";

    contact
      "Author: Bootstrap Admin <mailto:admin@example.com>";

    description
      "This module defines a data model to enable SZTP
       bootstrapping and discover what parameters are used.
       This module assumes the use of an IDevID certificate,
       as opposed to any other client certificate, or the
       use of an HTTP-based client authentication scheme.";

    revision 2019-01-15 {
      description
        "Initial version";
      reference
        "RFC XXXX: Secure Zero Touch Provisioning (SZTP)";
    }

    // features

    feature bootstrap-servers {
      description
        "The device supports bootstrapping off bootstrap servers.";
    }

    feature signed-data {
      description
        "The device supports bootstrapping off signed data.";
```

```
     }

     // protocol accessible nodes

     container sztp {
       description
         "Top-level container for SZTP data model.";
       leaf enabled {
         type boolean;
         default false;
         description
           "The 'enabled' leaf controls if SZTP bootstrapping is
            enabled or disabled.  The default is 'false' so that, when
            not enabled, which is most of the time, no configuration
            is needed.";
       }
       leaf idevid-certificate {
         if-feature bootstrap-servers;
         type ct:end-entity-cert-cms;
         config false;
         description
           "This CMS structure contains the IEEE 802.1AR-2009
            IDevID certificate itself, and all intermediate
            certificates leading up to, and optionally including,
            the manufacturer's well-known trust anchor certificate
            for IDevID certificates.  The well-known trust anchor
            does not have to be a self-signed certificate.";
         reference
           "IEEE 802.1AR-2009:
              IEEE Standard for Local and metropolitan area
              networks - Secure Device Identity.";
       }
       container bootstrap-servers {
         if-feature bootstrap-servers;
         config false;
         description
           "List of bootstrap servers this device will attempt
            to reach out to when bootstrapping.";
         list bootstrap-server {
           key "address";
           description
             "A bootstrap server entry.";
           leaf address {
             type inet:host;
             mandatory true;
             description
               "The IP address or hostname of the bootstrap server the
                device should redirect to.";
```

```
          }
          leaf port {
            type inet:port-number;
            default "443";
            description
              "The port number the bootstrap server listens on.  If no
               port is specified, the IANA-assigned port for 'https'
               (443) is used.";
          }
        }
      }
      container bootstrap-server-trust-anchors {
        if-feature bootstrap-servers;
        config false;
        description "Container for a list of trust anchor references.";
        leaf-list reference {
          type ta:pinned-certificates-ref;
          description
            "A reference to a list of pinned certificate authority (CA)
             certificates that the device uses to validate bootstrap
             servers with.";
        }
      }
      container voucher-trust-anchors {
        if-feature signed-data;
        config false;
        description "Container for a list of trust anchor references.";
        leaf-list reference {
          type ta:pinned-certificates-ref;
          description
            "A reference to a list of pinned certificate authority (CA)
             certificates that the device uses to validate ownership
             vouchers with.";
        }
      }
    }
  }
```

Appendix B. Promoting a Connection from Untrusted to Trusted

The following diagram illustrates a sequence of bootstrapping activities that promote an untrusted connection to a bootstrap server to a trusted connection to the same bootstrap server. This enables a device to limit the amount of information it might disclose to an adversary hosting an untrusted bootstrap server.

```
                                                        +----------+
                                                        |Deployment|
                                                        | Specific |
   +------+                                             |Bootstrap |
   |Device|                                             | Server   |
   +------+                                             +----------+
       |                                                          |
       | 1. "HTTPS" Request ("signed-data-preferred", nonce)      |
       |---------------------------------------------------------->|
       | 2. "HTTPS" Response (signed redirect information)        |
       |<----------------------------------------------------------|
       |                                                          |
       |                                                          |
       | 3. HTTPS Request (os-name=xyz, os-version=123, etc.)     |
       |---------------------------------------------------------->|
       | 4. HTTPS Response (unsigned onboarding information       |
       |<----------------------------------------------------------|
       |                                                          |
```

The interactions in the above diagram are described below.

1. The device initiates an untrusted connection to a bootstrap server, as is indicated by putting "HTTPS" in double quotes above. It is still an HTTPS connection, but the device is unable to authenticate the bootstrap server's TLS certificate. Because the device is unable to trust the bootstrap server, it sends the "signed-data-preferred" input parameter, and optionally also the "nonce" input parameter, in the "get-bootstrapping-data" RPC. The "signed-data-preferred" parameter informs the bootstrap server that the device does not trust it and may be holding back some additional input parameters from the server (e.g., other input parameters, progress reports, etc.). The "nonce" input parameter enables the bootstrap server to dynamically obtain an ownership voucher from a MASA, which may be important for devices that do not have a reliable clock.

2. The bootstrap server, seeing the "signed-data-preferred" input parameter, knows that it can either send unsigned redirect information or signed data of any type. But, in this case, the bootstrap server has the ability to sign data and chooses to respond with signed redirect information, not signed onboarding information as might be expected, securely redirecting the device back to it again. Not displayed but, if the "nonce" input parameter was passed, the bootstrap server could dynamically connect to a download a voucher from the MASA having the nonce value in it. Details regarding a protocol enabling this integration is outside the scope of this document.

3. Upon validating the signed redirect information, the device establishes a secure connection to the bootstrap server. Unbeknownst to the device, it is the same bootstrap server it was connected to previously but, because the device is able to authenticate the bootstrap server this time, it sends its normal "get-bootstrapping-data" request (i.e., with additional input parameters) as well as its progress reports (not depicted).

4. This time, because the "signed-data-preferred" parameter was not passed, having access to all of the device's input parameters, the bootstrap server returns, in this example, unsigned onboarding information to the device. Note also that, because the bootstrap server is now trusted, the device will send progress reports to the server.

Appendix C. Workflow Overview

The solution presented in this document is conceptualized to be composed of the non-normative workflows described in this section. Implementation details are expected to vary. Each diagram is followed by a detailed description of the steps presented in the diagram, with further explanation on how implementations may vary.

C.1. Enrollment and Ordering Devices

The following diagram illustrates key interactions that may occur from when a prospective owner enrolls in a manufacturer's SZTP program to when the manufacturer ships devices for an order placed by the prospective owner.

```
                                  +-----------+
   +------------+                 |Prospective|                     +---+
   |Manufacturer|                 |   Owner   |                     |NMS|
   +------------+                 +-----------+                     +---+
         |                              |                              |
         |                              |                              |
         |  1. initiate enrollment      |                              |
         #<-----------------------------|                              |
         #                              |                              |
         #                              |                              |
         #     IDevID trust anchor      |                              |
         #----------------------------->#  set IDevID trust anchor    |
         #                              #--------------------------->|
         #                              |                              |
         #     bootstrap server         |                              |
         #     account credentials      |                              |
         #----------------------------->#  set credentials            |
         |                              #--------------------------->|
         |                              |                              |
         |                              |                              |
         |  2. set owner certificate trust anchor                      |
         <-------------------------------------------------------------|
         |                              |                              |
         |                              |                              |
         |  3. place device order       |                              |
         <-----------------------------#  model devices               |
         |                              #--------------------------->|
         |                              |                              |
         |  4. ship devices and send    |                              |
         |     device identifiers and   |                              |
         |     ownership vouchers       |                              |
         |----------------------------->#  set device identifiers     |
         |                              #  and ownership vouchers      |
         |                              #--------------------------->|
         |                              |                              |
```

Each numbered item below corresponds to a numbered item in the diagram above.

1. A prospective owner of a manufacturer's devices initiates an enrollment process with the manufacturer. This process includes the following:

   * Regardless how the prospective owner intends to bootstrap their devices, they will always obtain from the manufacturer the trust anchor certificate for the IDevID certificates. This certificate will is installed on the prospective owner's

NMS so that the NMS can authenticate the IDevID certificates when they are presented to subsequent steps.

* If the manufacturer hosts an Internet based bootstrap server (e.g., a redirect server) such as described in Section 4.4, then credentials necessary to configure the bootstrap server would be provided to the prospective owner. If the bootstrap server is configurable through an API (outside the scope of this document), then the credentials might be installed on the prospective owner's NMS so that the NMS can subsequently configure the manufacturer-hosted bootstrap server directly.

2. If the manufacturer's devices are able to validate signed data (Section 5.4), and assuming that the prospective owner's NMS is able to prepare and sign the bootstrapping data itself, the prospective owner's NMS might set a trust anchor certificate onto the manufacturer's bootstrap server, using the credentials provided in the previous step. This certificate is the trust anchor certificate that the prospective owner would like the manufacturer to place into the ownership vouchers it generates, thereby enabling devices to trust the owner's owner certificate. How this trust anchor certificate is used to enable devices to validate signed bootstrapping data is described in Section 5.4.

3. Some time later, the prospective owner places an order with the manufacturer, perhaps with a special flag checked for SZTP handling. At this time, or perhaps before placing the order, the owner may model the devices in their NMS, creating virtual objects for the devices with no real-world device associations. For instance the model can be used to simulate the device's location in the network and the configuration it should have when fully operational.

4. When the manufacturer fulfills the order, shipping the devices to their intended locations, they may notify the owner of the devices' serial numbers and shipping destinations, which the owner may use to stage the network for when the devices power on. Additionally, the manufacturer may send one or more ownership vouchers, cryptographically assigning ownership of those devices to the owner. The owner may set this information on their NMS, perhaps binding specific modeled devices to the serial numbers and ownership vouchers.

C.2. Owner Stages the Network for Bootstrap

The following diagram illustrates how an owner might stage the network for bootstrapping devices.

```
                +----------+ +------------+
                |Deployment| |Manufacturer| +------+ +------+
                | Specific | |   Hosted   | | Local| | Local| +---------+
          +---+ |Bootstrap | | Bootstrap  | |  DNS | | DHCP | |Removable|
          |NMS| |  Server  | |   Server   | |Server| |Server| | Storage |
          +---+ +----------+ +------------+ +------+ +------+ +---------+
            |        |             |            |        |          |
   1.       |        |             |            |        |          |
   activate |        |             |            |        |          |
   modeled  |        |             |            |        |          |
   device   |        |             |            |        |          |
   ------->|        |             |            |        |          |
            | 2. (optional)        |            |        |          |
            |    configure         |            |        |          |
            |    bootstrap         |            |        |          |
            |    server            |            |        |          |
            |------->|             |            |        |          |
            |        |             |            |        |          |
            | 3. (optional) configure           |        |          |
            |    bootstrap server  |            |        |          |
            |--------------------->|            |        |          |
            |        |             |            |        |          |
            | 4. (optional) configure DNS server|        |          |
            |---------------------------------->|        |          |
            |        |             |            |        |          |
            | 5. (optional) configure DHCP server        |          |
            |------------------------------------------->|          |
            |        |             |            |        |          |
            | 6. (optional) store bootstrapping artifacts on media  |
            |------------------------------------------------------>|
            |        |             |            |        |          |
```

Each numbered item below corresponds to a numbered item in the diagram above.

1. Having previously modeled the devices, including setting their fully operational configurations and associating device serial numbers and (optionally) ownership vouchers, the owner might "activate" one or more modeled devices. That is, the owner tells the NMS to perform the steps necessary to prepare for when the real-world devices power up and initiate the bootstrapping process. Note that, in some deployments, this step might be combined with the last step from the previous workflow. Here it

is depicted that an NMS performs the steps, but they may be performed manually or through some other mechanism.

2. If it is desired to use a deployment-specific bootstrap server, it must be configured to provide the bootstrapping data for the specific devices. Configuring the bootstrap server may occur via a programmatic API not defined by this document. Illustrated here as an external component, the bootstrap server may be implemented as an internal component of the NMS itself.

3. If it is desired to use a manufacturer hosted bootstrap server, it must be configured to provide the bootstrapping data for the specific devices. The configuration must be either redirect or onboarding information. That is, either the manufacturer hosted bootstrap server will redirect the device to another bootstrap server, or provide the device with the onboarding information itself. The types of bootstrapping data the manufacturer hosted bootstrap server supports may vary by implementation; some implementations may only support redirect information, or only support onboarding information, or support both redirect and onboarding information. Configuring the bootstrap server may occur via a programmatic API not defined by this document.

4. If it is desired to use a DNS server to supply bootstrapping data, a DNS server needs to be configured. If multicast DNS-SD is desired, then the DNS server must reside on the local network, otherwise the DNS server may reside on a remote network. Please see Section 4.2 for more information about how to configure DNS servers. Configuring the DNS server may occur via a programmatic API not defined by this document.

5. If it is desired to use a DHCP server to supply bootstrapping data, a DHCP server needs to be configured. The DHCP server may be accessed directly or via a DHCP relay. Please see Section 4.3 for more information about how to configure DHCP servers. Configuring the DHCP server may occur via a programmatic API not defined by this document.

6. If it is desired to use a removable storage device (e.g., USB flash drive) to supply bootstrapping data, the data would need to be placed onto it. Please see Section 4.1 for more information about how to configure a removable storage device.

C.3. Device Powers On

The following diagram illustrates the sequence of activities that occur when a device powers on.

```
                                                   +----------+
                                     +-----------+ |Deployment|
                                     | Source of | | Specific |
   +------+                          | Bootstrap | |Bootstrap |  +---+
   |Device|                          |   Data    | |  Server  |  |NMS|
   +------+                          +-----------+ +----------+  +---+
       |                                   |            |          |
       |                                   |            |          |
       | 1. if SZTP bootstrap service      |            |          |
       |    is not enabled, then exit.     |            |          |
       |                                   |            |          |
       | 2. for each source supported, check |          |          |
       |    for bootstrapping data.        |            |          |
       |----------------------------------->|            |          |
       |                                   |            |          |
       | 3. if onboarding information found, |          |          |
       |    initialize self and, only if   |            |          |
       |    source is a trusted bootstrap  |            |          |
       |    server, send progress reports. |            |          |
       |----------------------------------->#            |          |
       |                                   # webhook    |          |
       |                                   #----------------------->|
       |                                                |          |
       | 4. else if redirect-information found, for each |         |
       |    bootstrap server specified, check for data. |          |
       |-+--------------------------------------------->|          |
       | |                                              |          |
       | | if more redirect-information is found, recurse |        |
       | | (not depicted), else if onboarding information |         |
       | | found, initialize self and post progress reports |       |
       | +----------------------------------------------->#         |
       |                                                # webhook  |
       |                                                #-------->|
       |
       | 5. retry sources and/or wait for manual provisioning.
       |
```

The interactions in the above diagram are described below.

1. Upon power being applied, the device checks to see if SZTP bootstrapping is configured, such as must be the case when running its "factory default" configuration. If SZTP bootstrapping is not configured, then the bootstrapping logic exits and none of the following interactions occur.

2. For each source of bootstrapping data the device supports, preferably in order of closeness to the device (e.g., removable

storage before Internet based servers), the device checks to see if there is any bootstrapping data for it there.

3. If onboarding information is found, the device initializes itself accordingly (e.g., installing a boot-image and committing an initial configuration). If the source is a bootstrap server, and the bootstrap server can be trusted (i.e., TLS-level authentication), the device also sends progress reports to the bootstrap server.

   * The contents of the initial configuration should configure an administrator account on the device (e.g., username, SSH public key, etc.), and should configure the device either to listen for NETCONF or RESTCONF connections or to initiate call home connections [RFC8071], and should disable the SZTP bootstrapping service (e.g., the "enabled" leaf in data model presented in Appendix A).

   * If the bootstrap server supports forwarding device progress reports to external systems (e.g., via a webhook), a "bootstrap-complete" progress report (Section 7.3) informs the external system to know when it can, for instance, initiate a connection to the device. To support this scenario further, the "bootstrap-complete" progress report may also relay the device's SSH host keys and/or TLS certificates, with which the external system can use to authenticate subsequent connections to the device.

   If the device successfully completes the bootstrapping process, it exits the bootstrapping logic without considering any additional sources of bootstrapping data.

4. Otherwise, if redirect information is found, the device iterates through the list of specified bootstrap servers, checking to see if the bootstrap server has bootstrapping data for the device. If the bootstrap server returns more redirect information, then the device processes it recursively. Otherwise, if the bootstrap server returns onboarding information, the device processes it following the description provided in (3) above.

5. After having tried all supported sources of bootstrapping data, the device may retry again all the sources and/or provide manageability interfaces for manual configuration (e.g., CLI, HTTP, NETCONF, etc.). If manual configuration is allowed, and such configuration is provided, the configuration should also disable the SZTP bootstrapping service, as the need for bootstrapping would no longer be present.

Appendix D. Change Log

D.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 Conveyed 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

D.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 Conveyed Information DHCP Option
- Added IANA request for media types for boot-image and configuration

D.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 manufacturer signed ownership vouchers and owner certificates.

- o Renamed configuration server to bootstrap server, a more representative name given the information devices download from it.

- o Replaced the concept of a configlet by defining a southbound interface for the bootstrap server using YANG.

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

D.4. 02 to 03

- o Minor update, mostly just to add an Editor's Note to show how this draft might integrate with the draft-pritikin-anima-bootstrapping-keyinfra.

D.5. 03 to 04

- o Major update formally introducing unsigned data and support for Internet-based redirect servers.

- o Added many terms to Terminology section.

- o Added all new "Guiding Principles" section.

- o Added all new "Sources for Bootstrapping Data" section.

- o Rewrote the "Interactions" section and renamed it "Workflow Overview".

D.6. 04 to 05

- o Semi-major update, refactoring the document into more logical parts

- o Created new section for information types

- o Added support for DNS servers

- o Now allows provisional TLS connections

- o Bootstrapping data now supports scripts

- o Device Details section overhauled

- o Security Considerations expanded

- o Filled in enumerations for notification types

D.7. 05 to 06

- Minor update
- Added many Normative and Informative references.
- Added new section Other Considerations.

D.8. 06 to 07

- Minor update
- Added an Editorial Note section for RFC Editor.
- Updated the IANA Considerations section.

D.9. 07 to 08

- Minor update
- Updated to reflect review from Michael Richardson.

D.10. 08 to 09

- Added in missing "Signature" artifact example.
- Added recommendation for manufacturers to use interoperable formats and file naming conventions for removable storage devices.
- Added configuration-handling leaf to guide if config should be merged, replaced, or processed like an edit-config/yang-patch document.
- Added a pre-configuration script, in addition to the post-configuration script from -05 (issue #15).

D.11. 09 to 10

- Factored ownership voucher and voucher revocation to a separate document: draft-kwatsen-netconf-voucher. (issue #11)
- Removed <configuration-handling> options "edit-config" and "yang-patch". (issue #12)
- Defined how a signature over signed-data returned from a bootstrap server is processed. (issue #13)

- o Added recommendation for removable storage devices to use open/standard file systems when possible. (issue #14)

- o Replaced notifications "script-[warning/error]" with "[pre/post]-script-[warning/error]". (goes with issue #15)

- o switched owner-certificate to be encoded using the PKCS #7 format. (issue #16)

- o Replaced md5/sha1 with sha256 inside a choice statement, for future extensibility. (issue #17)

- o A ton of editorial changes, as I went thru the entire draft with a fine-toothed comb.

D.12. 10 to 11

- o fixed yang validation issues found by IETFYANGPageCompilation. note: these issues were NOT found by pyang --ietf or by the submission-time validator...

- o fixed a typo in the yang module, someone the config false statement was removed.

D.13. 11 to 12

- o fixed typo that prevented Appendix B from loading the examples correctly.

- o fixed more yang validation issues found by IETFYANGPageCompilation. note: again, these issues were NOT found by pyang --ietf or by the submission-time validator...

- o updated a few of the notification enumerations to be more consistent with the other enumerations (following the warning/error pattern).

- o updated the information-type artifact to state how it is encoded, matching the language that was in Appendix B.

D.14. 12 to 13

- o defined a standalone artifact to encode the old information-type into a PKCS #7 structure.

- o standalone information artifact hardcodes JSON encoding (to match the voucher draft).

- o combined the information and signature PKCS #7 structures into a single PKCS #7 structure.

- o moved the certificate-revocations into the owner-certificate's PKCS #7 structure.

- o eliminated support for voucher-revocations, to reflect the voucher-draft's switch from revocations to renewals.

D.15. 13 to 14

- o Renamed "bootstrap information" to "onboarding information".

- o Rewrote DHCP sections to address the packet-size limitation issue, as discussed in Chicago.

- o Added Ian as an author for his text-contributions to the DHCP sections.

- o Removed the Guiding Principles section.

D.16. 14 to 15

- o Renamed action "notification" to "update-progress" and, likewise "notification-type" to "update-type".

- o Updated examples to use "base64encodedvalue==" for binary values.

- o Greatly simplified the "Artifact Groupings" section, and moved it as a subsection to the "Artifacts" section.

- o Moved the "Workflow Overview" section to the Appendix.

- o Renamed "bootstrap information" to "update information".

- o Removed "Other Considerations" section.

- o Tons of editorial updates.

D.17. 15 to 16

- o tweaked language to refer to "initial state" rather than "factory default configuration", so as accommodate white-box scenarios.

- o added a paragraph to Intro regarding how the solution primarily regards physical machines, but could be extended to VMs by a future document.

- o added a pointer to the Workflow Overview section (recently moved to the Appendix) to the Intro.

- o added a note that, in order to simplify the verification process, the "Conveyed Information" PKCS #7 structure MUST also contain the signing X.509 certificate.

- o noted that the owner certificate’s must either have no Key Usage or the Key Usage must set the "digitalSignature" bit.

- o noted that the owner certificate’s subject and subjectAltName values are not constrained.

- o moved/consolidated some text from the Artifacts section down to the Device Details section.

- o tightened up some ambiguous language, for instance, by referring to specific leaf names in the Voucher artifact.

- o reverted a previously overzealous s/unique-id/serial-number/ change.

- o modified language for when ZTP runs from when factory-default config is running to when ZTP is configured, which the factory-defaults should set .

D.18. 16 to 17

- o Added an example for how to promote an untrusted connection to a trusted connection.

- o Added a "query parameters" section defining some parameters enabling scenarios raised in last call.

- o Added a "Disclosing Information to Untrusted Servers" section to the Security Considerations.

D.19. 17 to 18

- o Added Security Considerations for each YANG module.

- o Reverted back to the device always sending its DevID cert.

- o Moved data tree to "get-bootstrapping-data" RPC.

- o Moved the "update-progress" action to a "report-progress" RPC.

- o Added an "signed-data-preferred" parameter to "get-bootstrapping-data" RPC.

- o Added the "ietf-zerotouch-device" module.

- o Lots of small updates.

D.20. 18 to 19

- o Fixed "must" expressions, by converting "choice" to a "list" of "image-verification", each of which now points to a base identity called "hash-algorithm". There's just one algorithm currently defined (sha-256). Wish there was a standard crypto module that could identify such identities.

D.21. 19 to 20

- o Now references I-D.ietf-netmod-yang-tree-diagrams.

- o Fixed tree-diagrams in Section 2 to always reflect current YANG (now they are now dynamically generated).

- o The "redirect-information" container's "trust-anchor" is now a CMS structure that can contain a chain of certificates, rather than a single certificate.

- o The "onboarding-information" container's support for image verification reworked to be extensible.

- o Added a reference to the "Device Details" section to the new example-device-data-model module.

- o Clarified that the device must always pass its IDevID certificate, even for untrusted bootstrap servers.

- o Fixed the description statement for the "script" typedef to refer to the [pre/post]-script-[warning/error] enums, rather than the legacy script-[warning/error] enums.

- o For the get-bootstrapping-data RPC's input, removed the "remote-id" and "circuit-id" fields, and added a "hw-model" field.

- o Improved DHCP error handling text.

- o Added MUST requirement for DHCPv6 client and server implementing [RFC3396] to handle URI lists longer than 255 octets.

- o Changed the "configuration" value in onboarding-information to be type "binary" instead of "anydata".

- o Moved everything from PKCS#7 to CMS (this shows up as a big change).

- o Added the early code point allocation assignments for the DHCP Options in the IANA Considerations section, and updated the RFC Editor note accordingly.

- o Added RFC Editor request to replace the assigned values for the CMS content types.

- o Relaxed auth requirements from device needing to always send IDevID cert to device needing to always send authentication credentials, as this better matches what RFC 8040 Section 2.5 says.

- o Moved normative module "ietf-zerotouch-device" to non-normative module "example-device-data-model".

- o Updated Title, Abstract, and Introduction per discussion on list.

D.22. 20 to 21

- o Now any of the three artifact can be encrypted.

- o Fixed some line-too-long issues.

D.23. 21 to 22

- o Removed specifics around how scripts indicate warnings or errors and how scripts emit output.

- o Moved the SZTP Device Data Model section to the Appendix.

- o Modified the YANG module in the SZTP Device Data Model section to reflect the latest trust-anchors and keystore drafts.

- o Modified types in other YANG modules to more closely emulate what is in draft-ietf-netconf-crypto-types.

D.24. 22 to 23

- o Rewrote section 5.6 (processing onboboarding information) to be clearer about error handling and retained state. Specifically:

  * Clarified that a script, upon having an error, must gracefully exit, cleaning up any state that might hinder subsequent executions.

  * Added ability for scripts to be executed again with a flag enabling them to clean up state from a previous execution.

  * Clarified that the conifguration commit is atomic.

  * Clarified that any error encountered after committing the configuration (e.g., in the "post-configuration-script") must rollback the configuration to the previous configuration.

  * Clarified that failure to successfully deliver the "bootstrap-initiated" and "bootstrap-complete" progress types must be treated as an error.

  * Clarified that "return to bootstrapping sequence" is to be interpreted in the recursive context. Meaning that the device rolls-back one loop, rather than start over from scratch.

- o Changed how a device verifies a boot-image from just "MUST match one of the supplied fingerprints" to also allow for the verification to use an cryptographic signature embedded into the image itself.

- o Added more "progress-type" enums for visibility reasons, enabling more strongly-typed debug information to be sent to the bootstrap server.

- o Added Security Considerations based on early SecDir review.

- o Added recommendation for device to send warning if the initial config does not disable the bootstrapping process.

D.25. 23 to 24

- o Follow-ups from SecDir and Shepherd.

- o Added "boot-image-complete" enumeration.

D.26. 24 to 25

- o Removed remaining old "bootstrapping information" term usage.

- o Fixed DHCP Option length definition.

- o Added reference to RFC 6187.

D.27. 25 to 26

- o Updated URI structure text (sec 8.3) and added norm. ref to RFC7230 reflecting Alexey Melnikov's comment.
- o Added IANA registration for the 'zerotouch' service, per IESG review from Adam Roach.
- o Clarified device's looping behavior and support for alternative provisioning mechanisms, per IESG review from Mirja Kuehlewind.
- o Updated "ietf-sztp-bootstrap-server:ssh-host-key" from leaf-list to list, per IESG review from Benjamin Kaduk.
- o Added option size text to DHCPv4 option size to address Suresh Krishnan's IESG review discuss point.
- o Updated RFC3315 to RFC8415 and associated section references.
- o Revamped the DNS Server section, after digging into Alexey Melnikov comment.
- o Fixed IETF terminology template section in both YANG modules.

D.28. 26 to 27

- o Added Security Consideration for cascading trust via redirects.
- o Modified the get-bootstrapping-data RPC's "nonce" input parameter to being a minimum of 16-bytes (used to be 8-bytes).
- o Added Security Consideration regarding possible reuse of device's private key.
- o Added Security Consideration regarding use of sign-then-encrypt.
- o Renamed "Zero Touch"/"zerotouch" throughout. Now uses "SZTP" when referring to the draft/solution, and "conveyed" when referring to the bootstrapping artifact.
- o Added missing text for "encrypted unsigned conveyed information" case.
- o Renamed "untrusted-connection" input paramter to "signed-data-preferred"
- o Switch yd:yang-data back to rc:yang-data

- o Added a couple features to the bootstrap-server module.

D.29. 27 to 28

- o Modified DNS section to no longer reference DNS-SD (now just plain TXT and SRV lookups, via multicast or unicast.
- o Registers "_sztp" in the DNS Underscore Global Scoped Entry Registry.
- o Updated 802.1AR reference to current spec version.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Michael Behringer, Dean Bogdanovic, Martin Bjorklund, Joe Clarke, Dave Crocker, Toerless Eckert, Stephen Farrell, Stephen Hanna, Wes Hardaker, David Harrington, Mirja Kuehlewind, Radek Krejci, Suresh Krishnan, Benjamin Kaduk, David Mandelberg, Alexey Melnikov, Russ Mundy, Reinaldo Penno, Randy Presuhn, Max Pritikin, Michael Richardson, Adam Roach, Phil Shafer, Juergen Schoenwaelder.

Special thanks goes to Steve Hanna, Russ Mundy, and Wes Hardaker for brainstorming the original solution during the IETF 87 meeting in Berlin.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Mikael Abrahamsson
T-Systems

EMail: mikael.abrahamsson@t-systems.se

Ian Farrer
Deutsche Telekom AG

EMail: ian.farrer@telekom.de

ANIMA WG
Internet-Draft
Intended status: Informational
Expires: August 17, 2015

M. Pritikin
M. Behringer
S. Bjarnason
Cisco

February 13, 2015

Bootstrapping Key Infrastructures
draft-pritikin-anima-bootstrapping-keyinfra-01

Abstract

This document specifies automated bootstrapping of an 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

Copyright Notice

Table of Contents

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:

- o New entity authentication: "Who is this? What is its identity?"

- o New entity authorization: "Is it mine? Do I want it? What are the chances it has been compromised?"

- o Domain authentication: "What is this domain’s claimed identity?"

- o 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 communiciations 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.

```
                                                   Factory components
                                                           .
                                                           . +------------+
                                                           . | Factory CA |
                                                           . +------------+
                                                           .        |
                                                           . +------------+
                                                           . |            |
      +--------------(provides)---------------------------|  Factory   |
      |                                       +---------->|            |
      |                                       |         . +------------+
      |                                       V         .
      |                               +---------------+  . +------------+
      |                               | Orchestrator  |  . | MASA       |
      V                               +---------------+  . | Service    |
   +-------+                                    |         . |            |
   | New   |        +------------+       +-----------+  . +------------+
   | Entity|<--L2-->|    Proxy   |<----->|           |  .......  ^
   |       |        +------------+       |           |           |
   |       |                             | Registrar |           |
   |       |                             |           |           |
   |       |<--DHCP-->(L3 bootstrap)     |           |           |
   |       |                             |           |           |
   |       |<-----L3--------------------( registrar )-----------+
   |       |                             ( may proxy )        |
   +-------+                             +-----------+
                                                   |
                            +----------------------------+
                  ^         |  Domain Certification      | ^
                  .         |      Authority             | .
                  .         +----------------------------+ .
                  .                                        .
                  ........................................
                                     |
                          "domain" components
```

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 deliminating the autonomic domain. See [I-D.irtf-nmrg-autonomic-network-definitions] for more information.

The overall flow is shown in Figure 2:

```
   +---------+               +----------+               +-----------+
   |  New    |               |          |               |  MASA     |
   | Entity  |               |  Domain  |               |  Service  |
   |         |               |          |               | (Internet)|
   +---------+               +----------+               +-----------+
       |                          |                           |
       |<-------discovery--------->|                           |
       |---802.1AR credential----->|                           |
       |                          |                           |
       |                    [ accept device? ]                |
       |                          |                           |
       |                          |---802.1AR identity-------->|
       |                          |---Domain ID--------------->|
       |                          |                           |
       |                          |                     [device belongs]
       |                          |                     [to domain?    ]
       |                          |                           |
       |                          |                  [update audit log]
       |                          |                           |
       |                          |<---device history log------|
       |                          |<-- authorization token-----|
       |                          |                           |
       |                  [ still accept device?]             |
       |                          |                           |
       |<----authorization token---|                           |
       |<----domain information----|                           |
       |                          |                           |
  [auth token valid?]             |                           |
       |                          |                           |
       |----domain enrolment------>|                           |
       |<----domain certificate----|                           |
       |                          |
```

Figure 1

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]]

ND [RFC2461] / [RFC4861] [[EDNOTE: TBD]] NOTE: Neighbor Discovery protocols do not describe a mechanism for forwarding messages.

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 assocaitions 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 boostrapping 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:

- o The New Entity is authenticated using the IEEE 802.1AR credentials. (EST support for .

- o 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:

- o the use of an IEEE 802.1AR IDevID credential,

- o 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:

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

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

- o 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 recieved. 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:

- o Device management.
- o Routing authentication.
- o 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:

- o 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").

- o 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.

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

- o 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 recieved 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 Registar 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 successfullly from the cilent 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:

```
"log":[
  {"date":"<date/time of the entry>"},
   "domainID":"<domainID as extracted from the root
                certificate within the PKCS7 of the
                authorization token request>",
   "nonce":"<any nonce if supplied (or NULL)>"},

  {"date":"<date/time of the entry>"},
   "domainID":"<domainID as extracted from the root
                certificate within the PKCS7 of the
                authorization token request>",
   "nonce":"<any nonce if supplied (or NULL)>"},
]
```

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 boostrapping 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

[IDevID] IEEE Standard, , "IEEE 802.1AR Secure Device Identifier", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7030] Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", RFC 7030, October 2013.

9.2. Informative References

[I-D.behringer-homenet-trust-bootstrap] Behringer, M., Pritikin, M., and S. Bjarnason, "Bootstrapping Trust on a Homenet", draft-behringer-homenet-trust-bootstrap-02 (work in progress), February 2014.

[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.

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com

ANIMA WG
Internet-Draft
Intended status: Informational
Expires: January 7, 2016

M. Pritikin
Cisco
M. Richardson
SSW
M. Behringer
S. Bjarnason
Cisco
July 6, 2015

Bootstrapping Key Infrastructures
draft-pritikin-anima-bootstrapping-keyinfra-02

Abstract

This document specifies automated bootstrapping of an 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

Copyright Notice

Table of Contents

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:

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

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).

drop ship The physical distribution of equipment containing the "factory default" configuration to a final destination. In zero-touch scenarios there is no staging or pre-configuration during drop-ship.

imprint the process where a device that wishes to join a network acquires it’s domain specific identity. This term is taken from Konrad Lorenz’s work in biology with new ducklings: during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their mother. [imprinting]

pledge the prospective device, which has the identity provided to at the factory. Neither the device nor the network knows if the device yet knows if this device belongs with this network. This is definition 6, according to [pledge]

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.

```
                                                        Vendor components
                                                       .
                                                       .+---------------+
      +--------------Drop Ship------------------------.| Manufacturer  |
      |                                                .+---------------+
      |                                                .| M anufacturer |
      |                                                .| A uthorized   |
      |                                                .| S igning      |
      |                                                .| A uthority    |
      |                                                .+---------------+
      V                                                ......   ^
   +-------+                                                    |
   | New   |        +------------+       +-----------+         |
   | Entity|<--L2-->|    Proxy   |<----->|           |         |
   |       |        +------------+       |           |         |
   |       |                             | Registrar |         |
   |       |                             |           |         |
   |       |<-----L3--------------------( may proxy )---------+
   |       |                             +-----------+
   |       |                                   |
   |       |                 +----------------------------+
   |       |<-----Enroll---->| Domain Certification       | ^
   |       |<-----Config---->|      Authority             | .
   +-------+        .        | Management and etc         | .
                    .        +----------------------------+ .
                    .                                       .
                    ........................................
                                    "domain" components
```

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.

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. The MASA is recommended to provide ownership validation services which allows for fully secure zero-touch bootstrap of domain certificates with mutual authentication.

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.

This document 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. Although manual deployment is not scalable and is not a focus of this document the necessary mechanisms are called out in this document to ensure all such edge conditions are covered by the architectural and protocol models.

3. Functional Overview

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

This section details the state machine and operational flow for each of the main three entities. The New Entity, the Domain (primarily the Registrar) and the MASA service.

The overall flow is shown in Figure 2:

```
   +---------+               +----------+               +-----------+
   |  New    |               |          |               |  MASA     |
   | Entity  |               |  Domain  |               |  Service  |
   |         |               |          |               | (Internet)|
   +---------+               +----------+               +-----------+
       |                          |                           |
       |<-------discovery--------->|                           |
       |---802.1AR credential----->|                           |
       |                          |                           |
       |                    [ accept device? ]                |
       |                          |                           |
       |                          |---802.1AR identity-------->|
       |                          |---Domain ID--------------->|
       |                          |                           |
       |                          |                     [device belongs]
       |                          |                     [to domain?    ]
       |                          |                           |
       |                          |                  [update audit log]
       |                          |                           |
       |                          |<---device history log------|
       |                          |<-- authorization token-----|
       |                          |                           |
       |                  [ still accept device?]             |
       |                          |                           |
       |<----authorization token---|                           |
       |<----config information----|                           |
       |                          |                           |
  [authorization token valid?]     |                           |
  [apply config information]       |                           |
       |                          |                           |
       |----domain enrolment------>|                           |
       |<----domain certificate----|                           |
       |                          |                           |
```

Figure 2

3.1. Behavior of a new entity

A New Entity that has not yet been bootstrapped attempts to find a local domain and join it.

States of a New Entity are as follows:

```
                +--------------+
                |   Start      |
                |              |
                +------+-------+
                       |
                +------v-------+
                |  Discover    |
   +------------>              |
   |            +------+-------+
   |                   |
   |            +------v-------+
   |            |  Identity    |
   ^------------+              |
   | rejected   +------+-------+
   |                   |
   |            +------v-------+
   |            |  Imprint     |   Optional
   ^------------+              <--+Manual input
   | Bad MASA   +------+-------+
   | response          |
   |            +------v-------+
   |            |  Enroll      |
   ^------------+              |
   | Enroll     +------+-------+
   | Failure           |
   |            +------v-------+
   |            |  Being       |
   ^------------+  Managed     |
    Factory     +--------------+
    reset
```

Figure 3

State descriptions are 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. Identify itself. This is done by presenting an 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. Imprint on the Registrar. This requires verification of 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. Enroll by accepting the domain specific information from the registrar, and by enrolling a domain certificate from the registrar using a standard enrollment protocol, e.g. Enrolment over Secure Transport (EST) [RFC7030].
5. The New Entity is now a member of and Being Managed by 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.

3.1.1. Discovery and Identity

Existing architectures provide the functionality for discovery of the Domain Registrar. Use of an existing architecture is preferred over development of a new architecture. Discovering of a Domain Proxy that facilitates communication through to the Domain Registrar is simplified as "discovery of the domain". A proxy is included in Figure 1 although the simplified flow in Figure 2 does not include a proxy - under the assuption that the proxy forwarding is mostly transparent to the New Entity. Existing architectures for investigation include:

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]]

ND [RFC2461] / [RFC4861] [[EDNOTE: TBD]] NOTE: Neighbor Discovery protocols do not describe a mechanism for forwarding messages.

Each provides a method for the New Entity to discover and initiate communication with a local neighbor which is assumed to be a member of the domain infrastructure. 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 assocaitions established are between the new device and the Registrar regardless of proxy operations. [[EDNOTE: this is the simplest and most direct threat model but should be evaluated against the anima use cases. It may be preferable to engage in secure communications with the proxy itself?]]

The New Entity is expected to identify itself during one of the communication protocol exchanges. For example using EAP-TLS. If the client identity is rejected the New Entity repeats the Discovery process using the next proxy or discovery method available. 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 using an appropriate L3 protocol mechanisms.

[[EDNOTE: it is unclear yet if discovery happens on a per interface basis or once per device. What is the requirement around joining multiple domains; is this a bootstrapping requirement or is this a broader autonomic requirement]]

3.1.2. Imprint

The domain trust anchor is received by the New Entity during the boostrapping protocol methods in the form of a MASA authorization token containing the domainID. The goal of the imprint state is to securely obtain a copy of this trust anchor without involving human interaction.

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

- o using the Implicit Trust Anchor database (not an autonomic solution because the URL must be securely distributed),
- o engaging a human user to authorize the CA certificate using out-of-band data (not an autonomic solution because the human user is involved),
- o using a configured Explicit TA database (not an autonomic solution because the distribution of an explicit TA database is not autonomic),

- o and using a Certificate-Less TLS mutual authentication method (not an autonomic solution because the distribution of symmetric key material is not autonomic).

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.

An arbitrary basic configuration information package that is signed by the domain can be delivered alongside the authorization token. This information is signed by the domain private keys and is a one time delivery containing information such as which enrollment server to communicate with and which management system to communicate with. It is intended as a limited basic configuration for these purposes and is not intended to deliver entire final configuration to the device.

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

3.1.3. 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:

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

3.1.4. Being Managed

Functionality to provide generic "configuration" information 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. This ensures that all communications with management systems which can divulge local security information (e.g. network topology or raw key material) is secured using the local credentials

issued during enrollment.

See Section 3.5.

3.2. 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 3.1.1, already provide this functionality nothing additional is defined.

3.3. 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:

```
   Contacted by New Entity
           +
           |
   +-------v----------+
   | Entity           | fail?
   | Authentication   +---------+
   +-------+----------+         |
           |                    |
   +-------v----------+         |
   | Entity           | fail?   |
   | Authorization    +--------->
   +-------+----------+         |
           |                    |
   +-------v----------+         |
   | Claiming the     | fail?   |
   | Entity           +--------->
   +-------+----------+         |
           |                    |
   +-------v----------+         |
   | Log Verification | fail?   |
   |                  +--------->
   +-------+----------+         |
           |                    |
   +-------v----------+    +----v-------+
   | Forward          |    |            |
   | Authorization    |    | Reject     |
   | token + config   |    | Device     |
   | to the Entity    |    |            |
   +------------------+    +------------+
   Figure 4
```

3.3.1. Entity Authentication

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

- o the use of an IEEE 802.1AR IDevID credential,
- o 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).

3.3.2. Entity Authorization

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:

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

3.3.3. 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 3.1.2 a New Entity will only proceed with bootstrapping if a validated MASA authorization token has been recieved. The New Entity therefore enforces that bootstrapping only occurs if the claim has been logged.

3.3.4. Log Verification

The Registrar requests the log information for the new entity from the MASA service. The log is verified to confirm that the following is true to the satisfaction of the registrar’s configured parameters:

- o Any nonceless entries in the log are associated with domainIDs recognized by the registrar. The registar MAY be configured to ignore the history of the device but it is RECOMMENDED that this only be configured if the MASA server is known to perform ownership validation or if Trusted Computing Group secure boot and remote attestation is available.
- o Any nonce’d entries are older than when the domain is known to have physical possession of the new entity or that the domainIDs are recognized by the registrar.

If any of these criteria are unacceptable to the registrar the entity is rejected.

3.3.5. Forwarding Authorization Token plus Configuration

The Registrar forwards the received authorization token to the new entity. To simplify the message flows an initial configuration package can be delivered at this time which is signed by a representative of the domain.

[[EDNOTE: format TBD. The configuration package signature data must contain the full certificate path sufficient for the new entity to use the domainID information (as a trust anchor) to accept and validate the configuration)]]

3.4. 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:

3.4.1. 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.

3.4.2. 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.

3.5. 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:
- o Device management.
- o Routing authentication.
- o Service discovery.

3.5.1. 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.

4. Domain Operator Activities

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

4.1. 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.

4.2. 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.

4.3. Accepting New Entities

For each New Entity the Registrar is informed of 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 asserts its unique identity to the registrar. The registrar accepts all devices without authorization checks. This mode does not provide security against intruders and is not recommended.
2. Per device acceptance: The new device asserts its unique identity to the registrar. 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 acceptance: In larger networks, neither of the previous approaches is acceptable. Default acceptance is not secure, and a manual per device methods do 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 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 using the per device acceptance methods.

4. Automated Whitelist: an automated process that builds the necessary whitelists and inserts them into the larger network domain infrastructure is plausible. Once set up, no human intervention is required in this process. Defining the exact mechanisms for this is out of scope although the registrar authorization checks is identified as the logical integration point of any future work in this area.

None of these approaches require 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 such that devices can enrol later. This supports use cases where the domain network may be entirely isolated during device deployment.

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

4.4. 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.

4.5. Secure Network Operations

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 3.5.1. 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.

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:

- o 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").
- o 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.
- o The New Entity calls the EST defined /cacerts method to obtain the current CA certificate. These are validated using the "bootstrap token".
- o 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" [[EDNOTE: out of date xref]] 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 3.3.2. If authorization is successful the Registrar obtains a MASA authorization token from the MASA service (see Section 5.3).

The recieved 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 Registar 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: The authorization token response format needs to be defined here. It consists of the nonce, if supplied, the serialnumber and the trust anchor of the domain. For example:

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

]]

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

5.4. Basic Configuration Information Package

When the MASA authorization token is returned to the New Entity an arbitrary information package can be signed and delivered along side it. This is signed by the Domain Registar. The New Entity first verifies the MASA authorization token and, if it is valid, then uses

the domain’s TA to validate the Information Package.

[[EDNOTE: The package format to be specified here. Any signed format is viable and ideally one can simply be specified from netconf. The Registar knows the New Entity device type from the 802.1AR credential and so is able to determine the proper format for the configuration]]

5.5. 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:

```
"log":[
  {"date":"<date/time of the entry>"},
   "domainID":"<domainID as extracted from the root
                certificate within the PKCS7 of the
                authorization token request>",
   "nonce":"<any nonce if supplied (or NULL)>"},

  {"date":"<date/time of the entry>"},
   "domainID":"<domainID as extracted from the root
                certificate within the PKCS7 of the
                authorization token request>",
   "nonce":"<any nonce if supplied (or NULL)>"},
]
```

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 is to support less secure

operational modes for support specific use cases. The following sections detail specific ways that the New Entity, Registrar and MASA can be configured to run in a less secure mode for the indicated reasons.

6.1. New Entity security reductions

Although New Entity can choose to run in less secure modes this is MUST NOT be the default state because it permanently degrades the security for all other uses cases. When configured into lower security modes by a trusted administrator:

1. 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 or when local staging has pre-configured the New Entity with a known good Trust Anchor.
2. The device may be configured during staging or requested from the factory to not require the MASA service authorization token. An entity that does not validate the domain identity is inherently dangerous as it may have had malware installed on it by a man-in-the-middle. 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 boostrapping protocol exchange is TBD]]. This may occur when: The device Factory does not provide a MASA service.

6.2. Registrar security reductions

The Registrar can choose to accept devices using less secure methods. These methods are RECOMMENDED when low security models are needed as the security decisions are being made by the local administrator:

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. The registrar may request nonce-less authorization tokens from the MASA service. These tokens can then be transmitted to the Registrar and stored until they are needed during bootstrapping operations. This is for use cases where target network is protected by an air gap and therefore can not contact the MASA

service during New Entity deployment.

6.3. MASA security reductions

Lower security modes chosen by the MASA service effect all device deployments unless paired with strict device ownership validation, in which case these modes can be provided as additional features for specific customers. The MASA service can choose to run in less secure modes by:

1. Not enforcing that a Nonce is in the authorization token. This results in distribution of authorization tokens that never expire and effectly makes the Domain an always trusted entity to the New Entity during any subsequent bootstrapping attempts. That this occured is captured in the log information so that the Domain registrar can make appropriate security decisions when a new device joins the domain. This is useful to support use cases where Registrars might not be online during actual device deployment.
2. Not verifying ownership before responding with an authorization token. Doing so relieves the vendor providing MASA services from having to tracking ownership during shipping and supply chain. The registrar uses the log information as a defense in depth strategy to ensure that this does not occur unexpectedly. For example when purchasing used equipment a MASA response is necessary for autonomic provisioning but the greatest level of security is achieved when the MASA server is also performing ownership validation.

7. Security Considerations

In order to support a wide 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. Or would result in an invalid nonce being associated with a 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, would now have to 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. If the MASA server were to have allowed a significantly large number of claims this might become onerous to the MASA server which must maintain all the extra log entries. Ensuring the registar

is representative of a valid customer domain even without validating ownership helps to mitigate this.

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. Binding the logs via a]] 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, Brian Carpenter, Fuyu Eleven.

9. References

9.1. Normative References

[IDevID] IEEE Standard, "IEEE 802.1AR Secure Device Identifier", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7030] Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", RFC 7030, October 2013.

9.2. Informative References

[I-D.behringer-homenet-trust-bootstrap]
Behringer, M., Pritikin, M., and S. Bjarnason, "Bootstrapping Trust on a Homenet", draft-behringer-homenet-trust-bootstrap-02 (work in progress), February 2014.

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

[imprinting]
Wikipedia, "Wikipedia article: Imprinting", July 2015, <https://en.wikipedia.org/wiki/Imprinting_(psychology)>.

[pledge] Dictionary.com, "Dictionary.com Unabridged", July 2015, <http://dictionary.reference.com/browse/pledge>.

Appendix A. Editor notes

[[EDNOTE: This section is to capturing rough notes between editors and Anima Bootstrapping design team members. This entire section to be removed en masse before finalization]]

Change Discussion:

02 Moved sections for readability, Updated introduction, simplified functional overview to avoid distractions from optional elements, addressed updated security considerations, fleshed out state machines.

The following is a non-prioritized list of work items currently identified:

- o Continue to address gaps/opportunities highlighted by community work on bootstrappping. Refs: IETF92 "Survey of Security Bootstrapping", Aana Danping He, behcet Sarikaya. "NETCONF Zero Touch Update for ANIMA" https://www.ietf.org/proceedings/92/anima.html and "Bootstrapping Key Infrastructures", Pritikin, Behringer, Bjarnason
- o Intergrate "Ownership Voucher" as a valid optional format for the MASA response. So long as the issuance of this is logged and captured in the log response then the basic flow and threat model is substantially the same.

- o Attempt to re-use existing work as per the charter: Toerless notes: a) are existing [eap] options? or too complex? or doens't work? b) our own method (e.g. EAP-ANIMA c) if b then investigate using signaling protocol).
- o

Authors' Addresses

Max Pritikin
Cisco

Email: pritikin@cisco.com

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

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

Michael H. Behringer
Cisco

Email: mbehring@cisco.com

Steinthor Bjarnason
Cisco

Email: sbjarnas@cisco.com

ANIMA
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

M. Stenberg

March 5, 2015

Autonomic Distributed Node Consensus Protocol
draft-stenberg-anima-adncp-00

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

Copyright Notice

Table of Contents

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:

- o 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

DNCP-style bidirectional peering relationship between nodes, and therefore SHOULD NOT contain Node Connection TLV..

Such objective-specific messages should either define some transaction id scheme (TBD - should it be here), or include the request verbatim within the replies, if any.

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 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]
Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", draft-ietf-homenet-dncp-00 (work in progress), January 2015.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

10.2. Informative references

[I-D.ietf-homenet-prefix-assignment]
Pfister, P., Paterson, B., and J. Arkko, "Distributed Prefix Assignment Algorithm", draft-ietf-homenet-prefix-assignment-03 (work in progress), February 2015.

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

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