NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

K. Watsen
Juniper Networks
M. Abrahamsson
T-Systems

March 13, 2017

Zero Touch Provisioning for NETCONF or RESTCONF based Management
draft-ietf-netconf-zerotouch-13

Abstract

This draft presents a secure technique for establishing a NETCONF or RESTCONF connection between a newly deployed device, configured with just its factory default settings, and its deployment specific network management system (NMS).

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. Please note that no other RFC Editor instructions are specified anywhere else in this document.

This document contains references to other drafts in progress, both in the Normative References section, as well as in body text throughout. Please update the following references to reflect their final RFC assignments:

o I-D.ieft-netconf-netconf-client-server

o I-D.ietf-anima-bootstrapping-keyinfra

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

o "XXXX" --> the assigned 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 "2017-03-13" --> the publication date of this draft

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

o Appendix A. 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 to do installations is both cost prohibitive and does not scale.

This document defines a bootstrapping strategy enabling devices to securely obtain bootstrapping data with no installer input, beyond physical placement and connecting network and power cables. The ultimate goal of this document is to enable a secure NETCONF [RFC6241] or RESTCONF [RFC8040] connection to the deployment specific network management system (NMS).

1.1. Use Cases

- 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 off of.

- 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 off of. If

no such information is available, or the device is unable to use the information provided, it can then reach out to network just as it would for the remotely administered network use-case.

1.2. Terminology

This document uses the following terms:

Artifact: The term "artifact" is used throughout to represent the any of the three artifacts defined in Section 4. These artifacts collectively provide all the bootstrapping data a device needs.

Bootstrapping Data: The term "bootstrapping data" is used throughout this document to refer to the collection of data that a device may obtain from any source of bootstrapping data. Specifically, it refers to the artifacts defined in Section 4.

Bootstrap Information: The term "bootstrap information" is used herein to refer to one of the bootstrapping artifacts defined in Section 4. Specifically, bootstrap information is the bootstrapping data that guides a device to, for instance, install a specific boot-image and commit a specific configuration.

Bootstrap Server: The term "bootstrap server" is used within this document to mean any RESTCONF server implementing the YANG module defined in Section 10.3.

Device: The term "device" is used throughout this document to refer to the network element that needs to be bootstrapped. See Section 8 for more information about devices.

Initial Secure Device Identifier (IDevID): The term "IDevID" is defined in [Std-802.1AR-2009] as the secure device identifier (DevID) installed on the device by the manufacturer. This identifier is used in this document to enable a Bootstrap Server to securely identify and authenticate a device.

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.

Owner: See Rightful Owner.

Redirect Information: The term "bootstrap information" is used herein to refer to one of the bootstrapping artifacts defined in Section 4. Specifically, redirect information is the bootstrapping data that directs a device to connect to a bootstrap server.

Redirect Server: The term "redirect server" is used to refer to a subset of bootstrap servers that only returns redirect information. A redirect server is particularly useful when hosted by a manufacturer, to redirect devices to deployment-specific bootstrap servers.

Rightful Owner: The term "rightful owner" is used herein to refer to the person or organization that purchased or otherwise owns a device. Ownership is further described in Section 2.3.

Signed Data: The term "signed data" is used throughout to mean either redirect information or bootstrap information that has been signed by a device's rightful owner's private key.

Unsigned Data: The term "unsigned data" is used throughout to mean either redirect information or bootstrap information that has not been signed by a device's rightful owner's private key.

1.3. Requirements Language

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].

1.4. Tree Diagram Notation

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" (read-write) represents configuration data and "ro" (read-only) represents state data.

- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Guiding Principles

This section provides overarching principles guiding the solution presented in this document.

2.1. Trust Anchors

A trust anchor is used in cryptography to represent an entity in which trust is implicit and not derived. In public key infrastructure using X.509 certificates, a root certificate is the trust anchor, from which a chain of trust is derived. The solution presented in this document requires that all the entities involved (e.g., devices, bootstrap servers, NMSs) possess specific trust anchors in order to ensure mutual authentication throughout the zero touch bootstrapping process.

2.2. Conveying Trust

A device in its factory default state possesses a limited set of manufacturer specified trust anchors. In this document, there are two types of trust anchors of interest. The first type of trust anchor is used to authenticate a secure (e.g., HTTPS) connection to, for instance, a manufacturer-hosted Internet-based bootstrap server. The second type of trust anchor is used to authenticate manufacturer-signed data, such as the ownership voucher artifact described in Section 4.3.

Using the first type of trust anchor, trust is conveyed by the device first authenticating the server (e.g., a bootstrap server), and then by the device trusting that the server would only provide data that its rightful owner staged for it to find. Thereby the device can trust any information returned from the server.

Using the second type of trust anchor, trust is conveyed by the device first authenticating that an artifact has been signed by its rightful owner, and thereby can trust any information held within the artifact.

Notably, redirect information, as described in Section 3.1, may include more trust anchors, which illustrates another way in which trust can be conveyed.

2.3. Conveying Ownership

The ultimate goal of this document is to enable a device to establish a secure connection with its rightful owner's NMS. This entails the manufacturer being able to track who is the rightful owner of a device (not defined in this document), as well as an ability to convey that information to devices (defined in this document).

Matching the two ways to convey trust (Section 2.2), this document provides two ways to convey ownership, by using a trusted bootstrap server (Section 6.4) or by using an ownership voucher (Section 4.3).

When a device connects to a trusted bootstrap server, one that was preconfigured into its factory default configuration, it implicitly trusts that the bootstrap server would only provide data that its rightful owner staged for it to find. That is, ownership is conveyed by the administrator of the bootstrap server (e.g., a manufacturer) taking the onus of ensuring that only data configured by a device's rightful owner is made available to the device. With this approach, the assignment of a device to an owner is ephemeral, as the administrator can reassign a device to another owner at any time.

When a device is presented signed bootstrapping data, it can authenticate that its rightful owner provided the data by verifying the signature over the data using an additional artifact defined within this document, the ownership voucher. With this approach, ownership is conveyed by the manufacturer (or delegate) taking the onus of ensuring that the ownership vouchers it issues are accurate.

3. Types of Zero Touch Information

This document defines two types of information that devices access during the bootstrapping process. These information types are described in this section.

3.1. Redirect Information

Redirect information provides information to redirect a device to a bootstrap server. Redirect information encodes a list of bootstrap servers, each defined by its hostname or IP address, an optional port, and an optional trust anchor certificate.

Redirect information is YANG modeled data formally defined by the "redirect-information" grouping in the YANG module presented in

Section 9.3. This grouping has the tree diagram shown below. Please see Section 1.4 for tree diagram notation.

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

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 rightful 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 bootstrap server, which is possible when the redirect information includes the bootstrap server's trust anchor certificate. When a device is able to establish a secure connection to a bootstrap server, the bootstrapping data does not have to be signed in order to be trusted, as described in Section 2.2.

Untrusted redirect information is useful for directing a device to a bootstrap server where signed data has been staged for it to obtain. When the redirect information is untrusted, the device MUST discard any potentially included trust anchor certificates. When the redirect information is untrusted, a device MAY establish a provisional connection to any of the specified bootstrap servers. A provisional connection is accomplished by the device blindly accepting the bootstrap server's TLS certificate. In this case, the device MUST NOT trust the bootstrap server, and data provided by the bootstrap server MUST be signed for it to be of any use to the device.

How devices process redirect information is described more formally in Section 8.5.

3.2. Bootstrap Information

Bootstrap information provides all the data necessary for a device to bootstrap itself, in order to be considered ready to be managed (e.g., by an NMS). As defined in this document, this data includes information about a boot image the device MUST be running, an initial configuration the device MUST commit, and optional scripts that, if specified, the device MUST successfully execute.

Bootstrap information is YANG modeled data formally defined by the "bootstrap-information" grouping in the YANG module presented in Section 9.3. This grouping has the tree diagram shown below. Please see Section 1.4 for tree diagram notation.

```
  +--:(bootstrap-information)
     +--ro bootstrap-information
        +--ro boot-image
        |  +--ro name       string
        |  +--ro (hash-algorithm)
        |  |  +--:(sha256)
        |  |     +--ro sha256?    string
        |  +--ro uri*       inet:uri
        +--ro configuration-handling       enumeration
        +--ro pre-configuration-script?    script
        +--ro configuration?
        +--ro post-configuration-script?   script
```

Bootstrap information MUST be trusted for it to be of any use to a device. There is no option for a device to process untrusted bootstrap information.

Bootstrap 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 rightful owner. In all other cases, the bootstrap information is untrusted.

How devices process bootstrap information is described more formally in Section 8.6.

4. Artifacts

This document defines three artifacts that can be made available to devices while they are bootstrapping. As will be seen in Section 6, each source of bootstrapping information specifies a means for providing each of the artifacts defined in this section.

4.1. Zero Touch Information

The information artifact encodes the essential bootstrapping data for the device. This artifact is used to encode the redirect information and bootstrap information types discussed in Section 3.

The information artifact is a PKCS#7 SignedData structure, as specified by Section 9.1 of [RFC2315], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.

Regardless how the information artifact is conveyed, the PKCS#7 structure MUST contain JSON-encoded content conforming to the YANG module specified in Section 9.3.

When the information artifact is conveyed over an untrusted transport (Section 2.2), the PKCS#7 structure structure MUST also contain a 'signerInfo' structure, as described in Section 9.1 of [RFC2315], containing a signature generated over the content using the private key associated with the owner certificate (Section 4.2).

4.2. Owner Certificate

The owner certificate artifact is a certificate that is used to identify an 'owner' (e.g., an organization), as known to a trusted certificate authority. The owner certificate is signed by a trusted certificate authority (CA), whose certificate is placed into the ownership voucher (Section 4.3).

The owner certificate is used by a device to verify the signature attached to the information artifact (Section 4.1) that the device SHOULD have also received, as described in Section 5. In particular, the device verifies signature using the public key in the owner certificate over the content contained within the information artifact.

In order to validate the owner certificate, a device MUST verify that the owner certificate's certificate chain includes the certificate specified by the ownership voucher (Section 4.3) that the device SHOULD have also received, as described in Section 5, and the device MUST verify that owner certificate contains an identifier matching the one specified in the voucher and, for devices that insist on verifying certificate revocation status, the device MUST verify that the certificate has neither expired nor been revoked.

The owner certificate artifact is formally an unsigned PKCS #7 SignedData structure as specified by Section 9.1 in [RFC2315], encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.

The owner certificate artifact MUST contain the owner certificate itself and all intermediate certificates leading up to the trust anchor certificate specified in the ownership voucher. The owner certificate artifact MAY optionally include the trust anchor certificate.

Additionally, if needed by the device, the owner certificate artifact MAY also contain suitably fresh CRLs [RFC5280] and/or OCSP Responses [RFC6960].

4.3. 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 or delegate.

The ownership voucher is used by a device to verify the owner certificate (Section 4.2) that the device SHOULD have also received, as described in Section 5. In particular, the device verifies that the owner certificate's chain of trust includes the trusted certificate included in the voucher, and the device also verifies that the owner certificate contains an identifier matching the one specified in the voucher.

In order to validate the voucher, a device MUST verify that the voucher was signed by the private key associated with a trusted certificate known to the device in its factory default state, as described in Section 8.1, and the device MUST verify that the voucher includes the device's unique identifier (e.g., serial number) and, if the voucher contains an expiration date, the device MUST also verify that the voucher has not expired.

The ownership voucher artifact, including its encoding, is formally defined in [I-D.ietf-anima-voucher].

5. Artifact Groupings

Section 4 lists all the possible bootstrapping artifacts, but only certain groupings of these artifacts make sense to return in the various bootstrapping situations described in this document. The remainder of this section identifies these groupings to further clarify how the artifacts are used.

5.1. Unsigned Information

The first grouping of artifacts is for unsigned information. That is, when the information artifact (Section 4.1) has not been signed.

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

Conveying unsigned information entails communicating just one of the three artifacts listed in Section 4 as follows:

List of artifacts included in this grouping:
  - zero touch information (with no embedded signature)

5.2. Signed Information (without Revocations)

The second grouping of artifacts is for when the information artifact (Section 4.1) has been signed, without any revocation information.

Signed information is needed when the information is obtained from an untrusted source of bootstrapping data (Section 6) and yet it is desired that the device be able to trust the information (i.e. no provisional processing).

Revocation information may not need to be provided because, for instance, the device only uses revocation information obtained dynamically from Internet based resources. Another possible reason may be because the device does not have a reliable clock, and therefore the manufacturer decides to never revoke information (e.g., ownership assignments are forever).

Conveying signed information without revocation information entails communicating all three of the artifacts listed in Section 4 as follows:

List of artifacts included in this grouping:
- zero touch information (with an embedded signature)
- owner certificate (with no revocation structures)
- ownership voucher

5.3. Signed Information (with Revocations)

The third grouping of artifacts is for when the information artifact (Section 4.1) has been signed and also includes revocation information.

Signed information, as described above, is needed when the information is obtained from an untrusted source of bootstrapping data (Section 6) and yet it is desired that the device be able to trust the information (i.e. no provisional processing).

Revocation information may need to be provided because, for instance, the device insists on being able to verify revocations and the device is deployed on a private network and therefore unable to obtain the revocation information from Internet based resources.

Conveying signed information with revocation information entails communicating all three of the artifacts listed in Section 4 as follows:

List of artifacts included in this grouping:
- zero touch information (with an embedded signature)
- owner certificate (with revocation structures)
- ownership voucher

6. Sources of Bootstrapping Data

This section defines some sources for zero touch 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 zero touch bootstrapping data.

For each source defined in this section, details are given for how each of the three artifacts listed in Section 4 is provided.

6.1. Removable Storage

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

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.

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

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 it 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 4 are mapped to files below.

Artifact to File Mapping:

Information: Mapped to a file containing the binary artifact described in Section 4.1.

Owner Certificate: Mapped to a file containing the binary artifact described in Section 4.2.

Ownership Voucher: Mapped to a file containing the binary artifact described in Section 4.3.

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 to exist on a removable storage device. The file naming convention SHOULD be unique to the manufacturer, in order to enable bootstrapping data from multiple manufacturers to exist on a removable storage device.

6.2. DNS Server

A DNS server MAY be used as a source of zero touch 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.

To use a DNS server as a source of bootstrapping data, a device MAY perform a multicast DNS [RFC6762] query searching for the service "_zerotouch._tcp.local.". Alternatively the device MAY perform DNS-SD [RFC6763] via normal DNS operation, using the domain returned to it from the DHCP server; for example, searching for the service "_zerotouch._tcp.example.com".

Unsigned DNS records (not using DNSSEC as described in [RFC6698]) are an untrusted source of bootstrapping data. This means that the information stored in the DNS records either MUST be signed, or it MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a DNS server presents resource records (Section 3.2.1 of [RFC1035]), the bootstrapping artifacts need to be presented as resource records. The three artifacts defined in Section 4 are mapped to resource records below.

Artifact to Resource Record Mapping:

Information: Mapped to a TXT record called "zt-info" containing the base64-encoding of the binary artifact described in Section 4.1.

Owner Certificate: Mapped to a TXT record called "zt-cert" containing the base64-encoding of the binary artifact described in Section 4.2.

Ownership Voucher: Mapped to a TXT record called "zt-voucher" containing the base64-encoding of the binary artifact described in Section 4.3.

TXT records have an upper size limit of 65535 bytes (Section 3.2.1 in RFC1035), since 'RDLENGTH' is a 16-bit field. Please see Section 3.1.3 in RFC4408 for how a TXT record can achieve this size. Due to this size limitation, some information artifacts may not fit. In particular, the bootstrap information artifact could hit this upper bound, depending on the size of the included configuration and scripts.

When bootstrap information is provided, it is notable that the URL for the boot-image the device can download would have to point to another server (e.g., http://, ftp://, etc.), as DNS servers do not themselves distribute files.

6.3. DHCP Server

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

To use a DHCP server as a source of bootstrapping data, a device need only send a DHCP lease request to a DHCP server. However, the device SHOULD pass the Vendor Class Identifier (option 60) field in its DHCP lease request, so the DHCP server can return bootstrap information shared by devices from the same vendor. However, if it is desired to return device-specific bootstrap information, then the device SHOULD also send the Client Identifier (option 61) field in its DHCP lease request, so the DHCP server can select the specific bootstrap information that has been staged for that one device.

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. This means that the information returned by the DHCP server either MUST be signed, or it MUST be information that can be processed provisionally (e.g., unsigned redirect information).

From an artifact perspective, since a DHCP server presents data as DHCP options , the bootstrapping artifacts need to be presented as

DHCP options, specifically the ones specified in Section 12.1. The three artifacts defined in Section 4 are mapped to the DHCP options specified in Section 12.1 below.

Artifact to DHCP Option Field Mapping:

Information: Mapped to the DHCP option field "zerotouch-information" containing the binary artifact described in Section 4.1.

Owner Certificate: Mapped to the DHCP option field "owner-certificate" containing the binary artifact described in Section 4.2.

Ownership Voucher: Mapped to the DHCP option field "ownership-voucher" containing the binary artifact described in Section 4.3.

When bootstrap information is provided, it is notable that the URL for the boot-image the device can download would have to point to another server (e.g., http://, ftp://, etc.), as DHCP servers do not themselves distribute files.

6.4. Bootstrap Server

A bootstrap server MAY be used as a source of zero touch bootstrapping data. A bootstrap server is defined as a RESTCONF [RFC8040] server implementing the YANG module provided in Section 10.

Unlike any other source of bootstrap 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 "notification" action statement defined in the YANG module (Section 10.3). The data sent from devices both enables visibility into the bootstrapping process (e.g., warnings and errors) as well as provides potentially useful completion status information (e.g., the device's SSH host-keys).

To use a bootstrap server as a source of bootstrapping data, a device MUST use the RESTCONF protocol to access the YANG container node /device/, passing its own serial number in the URL as the key to the 'device' list.

Using a bootstrap server as a source of bootstrapping data is a compelling option as it uses transport-level security in lieu of signed data, which may be easier to deploy in some situations. Additionally, the bootstrap server is able to receive notifications

from devices, which may be critical to some deployments (e.g., the passing of the device's SSH host keys).

A bootstrap server may be trusted or an untrusted source of bootstrapping data, depending on how the device learned about the bootstrap server's trust anchor from a trusted source. When a bootstrap server is trusted, the information returned from it MAY be signed. However, when the server is untrusted, in order for its information to be of any use to the device, the information MUST either be signed or be information that can be processed provisionally (e.g., unsigned redirect information).

When a device is able to trust a bootstrap server, it MUST send its IDevID certificate in the form of a TLS client certificate, and it MUST send notifications to the bootstrap server. When a device is not able to trust a bootstrap server, it MUST NOT send its IDevID certificate in the form of a TLS client certificate, and it MUST NOT send any notifications to the bootstrap server.

From an artifact perspective, since a bootstrap server presents data as a YANG-modeled data, the bootstrapping artifacts need to be mapped to nodes in the YANG module. The three artifacts defined in Section 4 are mapped to bootstrap server nodes defined in Section 10.3 below.

Artifact to Bootstrap Server Node Mapping:

Information: Mapped to the node /device/zerotouch-information.

Owner Certificate: Mapped to the leaf node /device/owner-certificate.

Ownership Voucher: Mapped to the leaf node /device/ownership-voucher.

While RESTCONF servers typically support a nested hierarchy of resources, zero touch bootstrap servers only need to support the paths /device and /device/notification. The device processing instructions provided in Section 8.3 only uses these two URLs.

7. Workflow Overview

The zero touch solution presented in this document is conceptualized to be composed of the workflows described in this section. Implementations MAY vary in details. Each diagram is followed by a detailed description of the steps presented in the diagram, with further explanation on how implementations may vary.

7.1. Onboarding and Ordering Devices

The following diagram illustrates key interactions that may occur from when a prospective owner enrolls in a manufacturer's zero touch 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           |
         #                              #--------------------------->|
         #                              |                            |
         #                              |                            |
         #  owner certificate           |                            |
         #----------------------------->#  set certificate           |
         |                              #--------------------------->|
         |                              |                            |
         |                              |                            |
         |  2. place device order       |                            |
         |<-----------------------------#  model devices             |
         |                              #--------------------------->|
         |                              |                            |
         |  3. 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, or an existing owner that wishes to start using zero touch for future device orders, initiates an enrollment process with the manufacturer or delegate. This process includes the following:

   * Regardless how the prospective owner intends to bootstrap their devices, they will always obtain from the manufacturer or delegate the trust anchor certificate for its device's IDevID certificates. This certificate will need to be installed on the prospective owner's NMS so that the NMS can subsequently authenticate the device's IDevID certificates.

   * If the manufacturer hosts an Internet based bootstrap server (e.g., a redirect server) such as described in Section 6.4, 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.

   * If the manufacturer's devices are able to validate signed data (Section 8.4), then the manufacturer, acting as a certificate authority, may additionally sign an owner certificate for the prospective owner. Alternatively, and not depicted, the owner may obtain an owner certificate from a manufacturer-trusted 3rd-party certificate authority, and report that certificate to the manufacturer. How the owner certificate is used to enable devices to validate signed bootstrapping data is described in Section 8.4. Assuming the prospective owner's NMS is able to prepare and sign the bootstrapping data, the owner certificate would be installed on the NMS at this time.

2. Some time later, the prospective owner places an order with the manufacturer (or delegate), perhaps with a special flag checked for zero touch 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.

3. When the manufacturer or delegate fulfills the order, shipping the devices to their intended locations, they may notify the owner of the devices's unique identifiers (e.g., 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
       rightful owner. The owner may set this information on their NMS,
       perhaps binding specific modeled devices to the unique
       identifiers and ownership vouchers.

7.2. 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 |
          +---+ +----------+ +------------+ +------+ +------+ +---------+
            |        |             |            |        |         |
   activate |        |             |            |        |         |
   modeled  |        |             |            |        |         |
1. 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 both device identifiers (e.g., serial numbers) and ownership vouchers, the owner "activates" 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 information 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 (or delegate) hosted bootstrap server, it MUST be configured to provide the bootstrapping information for the specific devices. The configuration MUST be either redirect or bootstrap information. That is, either the manufacturer hosted bootstrap server will redirect the device to another bootstrap server, or provide the device with its bootstrapping information itself. The types of bootstrapping information the manufacturer hosted bootstrap server supports MAY vary by implementation; some implementations may only support redirect information, or only support bootstrap information, or support both redirect and bootstrap 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 information, a DNS server needs to be configured. If multicast DNS-SD is desired, then the server MUST reside on the local network, otherwise the DNS server MAY reside on a remote network. Please see Section 6.2 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 6.3 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 information, the information would need to be placed onto it. Please see Section 6.1 for more information about how to configure a removable storage device.

7.3. 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 running a modified (not        |              |         |
     |    factory default) configuration,   |              |         |
     |    then exit.                        |              |         |
     |                                      |              |         |
     | 2. for each source supported, check  |              |         |
     |------------------------------------->|              |         |
     |                                      |              |         |
     | 3. if bootstrap-information found,   |              |         |
     |    initialize self and, only if      |              |         |
     |    source is a bootstrap server,     |              |         |
     |    send notifications                |              |         |
     |------------------------------------->#              |         |
     |                                      # webhook      |         |
     |                                      #----------------------->|
     |                                                     |         |
     | 4. else if redirect-information found, for          |         |
     |    each bootstrap server specified, check           |         |
     |-+-------------------------------------------------->|         |
     | |                                                   |         |
     | |  if more redirect-information is found, recurse   |         |
     | |  (not depicted), else if bootstrap-information    |         |
     | |  found, initialize self and post notifications    |         |
     | +-------------------------------------------------->#         |
     |                                                     # 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's bootstrapping logic first checks to see if it is running in its factory default state. If it is in a modified state, 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 bootstrap-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 notifications to the bootstrap server.

   * The contents of the initial configuration SHOULD configure an administrator account on the device (e.g., username, ssh-rsa key, etc.) and SHOULD configure the device either to listen for NETCONF or RESTCONF connections or to initiate call home connections [RFC8071].

   * If the bootstrap server supports forwarding device notifications to external systems (e.g., via a webhook), the "bootstrap-complete" notification (Section 10.3) informs the external system to know when it can, for instance, initiate a connection to the device (assuming it knows the device's address and the device was configured to listen for connections). To support this further, the bootstrap-complete notification also relays 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 is ever able to complete the bootstrapping process successfully (i.e., no longer running its factory default configuration), 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 there is any bootstrapping data for it on them. If the bootstrap server returns more redirect-information, then the device processes it recursively. Otherwise, if the bootstrap server returns bootstrap-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 device MUST immediately cease trying to obtain bootstrapping data, as it would then no longer be in running its factory default configuration.

8. Device Details

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

8.1. Factory Default State

```
+------------------------------------------------------------------+
|                             <device>                             |
|                                                                  |
|   +----------------------------------------------------------+   |
|   |                   <read-only storage>                    |   |
|   |                                                          |   |
|   | 1. IDevID cert & associated intermediate certificate(s)  |   |
|   | 2. list of trusted Internet based bootstrap servers      |   |
|   | 3. list of trust anchor certs for bootstrap servers      |   |
|   | 4. trust anchor cert for ownership vouchers              |   |
|   +----------------------------------------------------------+   |
|                                                                  |
|                    +----------------------+                      |
|                    |   <secure storage>   |                      |
|                    |                      |                      |
|                    |  5. private key      |                      |
|                    +----------------------+                      |
|                                                                  |
+------------------------------------------------------------------+
```

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

1. 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 the device's unique device identifier (e.g., serial number). The device MUST also possess any intermediate certificates between the IDevID certificate and the manufacturer's IDevID trust anchor certificate, which is provided to prospective owners separately (e.g., Section 7.1).

2. Devices that support loading bootstrapping data from an Internet-based bootstrap server (see Section 6.4) MUST be manufactured with a configured list of trusted bootstrap servers. Consistent with redirect information (Section 3.1, each bootstrap server MAY be identified by its hostname or IP address, and an optional port.

3. Devices that support loading bootstrapping data from an Internet-based bootstrap server (see Section 6.4) MUST also be manufactured with a list of trust anchor certificates that can be used for X.509 certificate path validation ([RFC6125], Section 6) on the bootstrap server's TLS server certificate.

4. Devices that support loading owner signed data (see Section 1.2) MUST also be manufactured with the trust anchor certificate for the ownership vouchers.

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

8.2. 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. Running default config?   -------->  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 off any source?  -----> Run with new configuration
       |
       | No
       v
4. Loop and/or wait for manual provisioning.
```

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 it is running the factory default configuration. If it is running a modified configuration, then it boots normally.

2. The device iterates over its list of sources for bootstrapping data (Section 6). Details for how to processes a source of bootstrapping data are provided in Section 8.3.

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

4. Otherwise the device MAY loop back through the list of bootstrapping sources again and/or wait for manual provisioning.

8.3. Processing a Source of Bootstrapping Data

This section describes a recursive algorithm that a device claiming to support the bootstrapping strategy defined in this document MUST use to authenticate bootstrapping data. A device enters this algorithm for each new source of bootstrapping data. The first time the device enters this algorithm, it MUST initialize a conceptual trust state variable, herein referred to as "trust-state", to FALSE. The ultimate goal of this algorithm is for the device to process bootstrap information (Section 3.2) while the trust-state variable is TRUE.

If the data source is a bootstrap server, and the device is able to authenticate the 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.

If trust-state is TRUE, when connecting to the bootstrap server, the device MUST use its IDevID certificate for client certificate based authentication and MUST POST progress notifications using the bootstrap server's "notification" action. Otherwise, if trust-state is FALSE, when connecting to the bootstrap server, the device MUST NOT use its IDevID certificate for a client certificate based authentication and MUST NOT POST progress notifications using the bootstrap server's "notification" action.

When accessing a bootstrap server, the device MUST only access its top-level resource, to obtain all the data staged for it in one GET request, so that it can determine if the data is signed or not, and

thus act accordingly. If trust-state is TRUE, then the device MAY also accesses the bootstrap servers 'notification' resource for the device.

For any source of bootstrapping data (e.g., Section 6), if the data is signed and the device is able to validate the signed data using the algorithm described in Section 8.4, then the device MUST set trust-state to TRUE, else 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 bootstrap server.

If the data is bootstrap information (not redirect information), and trust-state is FALSE, the device MUST exit the recursive algorithm, returning to the state machine described in Section 8.2. Otherwise, the device MUST attempt to process the bootstrap information as described in Section 8.6. In either case, success or failure, the device MUST exit the recursive algorithm, returning to the state machine described in Section 8.2, the only difference being in how it responds to the "Able to bootstrap off any source?" conditional described in that state machine.

If the data is redirect information, the device MUST process the redirect information as described in Section 8.5. This is the recursion step, it will cause to device to reenter this algorithm, but this time the data source will most definitely be a bootstrap server, as that is all redirect information is able to do.

8.4. Validating Signed Data

Whenever a device is presented signed data from an untrusted source, it MUST validate the signed data as described in this section. If the signed data is provided by a trusted source, a redundant trust case, the device MAY skip verifying the signature.

Whenever there is signed data, the device MUST also be provided an ownership voucher and an owner certificate. Depending on circumstances, the device MAY also be provided certificate revocations. How all the needed artifacts are provided for each source of bootstrapping data is defined in Section 6.

The device MUST first authenticate the ownership voucher by validating the signature on it to one of its preconfigured trust anchors (see Section 8.1) and verify that the voucher contains the device's unique identifier (e.g., serial number). If the voucher contains an expiration timestamp, the device MUST also verify that the voucher has not expired. If the authentication of the voucher is successful, the device extracts from it information that can be used to verify the owner certificate in the next step.

Next the device MUST authenticate the owner certificate by performing X.509 certificate path verification to the trusted certificate provided in the voucher. If the device insists on verifying revocation status, it MUST also verify that none of the certificates in the chain of certificates have been revoked or expired. If the authentication of the certificate is successful, the device extracts the owner's public key from the certificate for use in the next step.

Finally the device MUST verify the signature over information artifact was generated by the private key matching the public key extracted from the owner certificate in the previous step.

If any of these steps fail, then the device MUST mark the data as invalid and not perform any of the subsequent steps.

8.5. Processing Redirect Information

In order to process redirect information (Section 3.1), 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 off of.

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 bootstrap server using X.509 certificate path validation ([RFC6125], Section 6) to the specified trust anchor. If the device is unable to authenticate the bootstrap server to the specified trust anchor, the device MUST NOT attempt a provisional connection to the bootstrap server (i.e., by blindly accepting its server certificate).

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).

8.6. Processing Bootstrap Information

In order to process bootstrap information (Section 3.2), the device MUST follow the steps presented in this section.

When processing bootstrap information, the device MUST first process the boot image information, then execute the pre-configuration script (if any), then commit the initial configuration, and then execute the script (if any), in that order. If the device encounters an error at any step, it MUST NOT proceed to the next step.

First the device MUST determine if the image it is running satisfies the specified boot image criteria (e.g., name or fingerprint match). If it does not, the device MUST download, verify, and install the specified boot image, and then reboot. To verify the boot image, the device MUST check that the boot image file matches the fingerprint (e.g., sha256) supplied by the bootstrapping information. Upon rebooting, the device MUST still be in its factory default state, causing the bootstrapping process to run again, which will eventually come to this very point, but this time the device's running image will satisfy the specified criteria, and thus the device will move to processing the next step.

Next, for devices that support executing scripts, if a pre-configuration script has been specified, the device MUST execute the script and check its exit status code to determine if had any warnings or errors. In the case of errors, the device MUST reset itself in such a way that force the reinstallation of its boot image, thereby wiping out any bad state the script might have left behind.

Next the device commits the provided initial configuration. Assuming no errors, the device moves to processing the next step.

Again, for devices that support executing scripts, if a post-configuration script has been specified, the device MUST execute the script and check its exit status code to determine if it had any warnings or errors. In the case of errors, the device MUST reset itself in such a way that force the reinstallation of its boot image, thereby wiping out any bad state the script might have left behind.

At this point, the device has completely processed the bootstrapping data and is ready to be managed. If the device obtained the bootstrap information from a trusted bootstrap server, the device MUST send the 'bootstrap-complete' notification now.

At this point the device is configured and no longer running its factory default configuration. Notably, if the bootstrap information

configured the device it initiate a call home connection, the device would proceed to do so now.

9. The Zero Touch Information Artifact

This section defines a YANG [RFC6020] module that is used to define the data model for the Zero Touch Information artifact described in Section 4.1. Examples illustrating this artifact in use are provided in Section 9.2.

9.1. Tree Diagram

The following tree diagram provides an overview of the data model for the Zero Touch Information artifact. The syntax used for this tree diagram is described in Section 1.4.

```
module: ietf-zerotouch-information
  +---- (information-type)
     +--:(redirect-information)
     |  +---- redirect-information
     |     +---- bootstrap-server* [address]
     |        +---- address         inet:host
     |        +---- port?           inet:port-number
     |        +---- trust-anchor?   binary
     +--:(bootstrap-information)
        +---- bootstrap-information
           +---- boot-image
           |  +---- name      string
           |  +---- (hash-algorithm)
           |  |  +--:(sha256)
           |  |     +---- sha256?   string
           |  +---- uri*      inet:uri
           +---- configuration-handling?      enumeration
           +---- pre-configuration-script?    script
           +---- configuration?
           +---- post-configuration-script?   script
```

9.2. Example Usage

This section presents examples for how the information artifact (Section 4.1) can be encoded into a document that can be distributed outside the bootstrap server's RESTCONF API.

The following example illustrates how redirect information can be encoded into an artifact.

```
<redirect-information
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-information">
  &lt;bootstrap-server&gt;
    &lt;address&gt;phs1.example.com&lt;/address&gt;
    &lt;port&gt;8443&lt;/port&gt;
    &lt;trust-anchor&gt;Base64-encoded X.50 &lt;/trust-anchor&gt;
  &lt;/bootstrap-server&gt;
  &lt;bootstrap-server&gt;
    &lt;address&gt;phs2.example.com&lt;/address&gt;
    &lt;port&gt;8443&lt;/port&gt;
    &lt;trust-anchor&gt;Base64-encoded X.50 &lt;/trust-anchor&gt;
  &lt;/bootstrap-server&gt;
  &lt;bootstrap-server&gt;
    &lt;address&gt;phs3.example.com&lt;/address&gt;
    &lt;port&gt;8443&lt;/port&gt;
    &lt;trust-anchor&gt;Base64-encoded X.50 &lt;/trust-anchor&gt;
  &lt;/bootstrap-server&gt;
&lt;/redirect-information&gt;
```

The following example illustrates how bootstrap information can be encoded into an artifact. This example uses datamodels from [RFC7317] and [I-D.ietf-netconf-netconf-client-server].

```
<-- '\' line wrapping added for formatting purposes only -->

<bootstrap-information
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-information">
  &lt;boot-image&gt;
    &lt;name&gt;boot-image-v3.2R1.6.img&lt;/name&gt;
    &lt;sha256&gt;Hex-encoded SHA256 hash&lt;/sha256&gt;
    &lt;uri&gt;file:///some/path/to/raw/file &lt;/uri&gt;
  &lt;/boot-image&gt;
  &lt;configuration-handling&gt;merge&lt;/configuration-handling&gt;
  &lt;configuration&gt;
    &lt;!-- from ietf-system.yang --&gt;
    &lt;system xmlns="urn:ietf:params:xml:ns:yang:ietf-system"&gt;
      &lt;authentication&gt;
        &lt;user&gt;
          &lt;name&gt;admin&lt;/name&gt;
          &lt;authorized-key&gt;
            &lt;name&gt;admin's rsa ssh host-key&lt;/name&gt;
            &lt;algorithm&gt;ssh-rsa&lt;/algorithm&gt;
            &lt;key-data&gt;AAAAB3NzaC1yc2EAAAADAQABAAABAQDeJMV8zrtsi8CgEsRC\
            jCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/O2Mwj\
            E1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcC\
            WAw1lOr9IDGDAuww6G45gLcHalHMmBtQxKnZdzU9kx/fL3ZS5G76Fy6sA5\
            vg7SLqQFPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2DKqBIe340jWq\
```

```
            EIuA7LvEJYql4unq4Iog+/+CiumTkmQIWRgIoj4FCzYkO9NvRE6fOSLLf6\
            gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</key-data>
          </authorized-key>
        </user>
      </authentication>
    </system>
    <!-- from ietf-netconf-server.yang -->
    <netconf-server
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
      <call-home>
        <netconf-client>
          <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>
                <name>certificate</name>
                <certificate>builtin-idevid-cert</certificate>
              </host-key>
            </host-keys>
            <client-cert-auth>
              <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
              <trusted-client-certs>explicitly-trusted-client-certs</trusted-
client-certs>
            </client-cert-auth>
          </ssh>
          <connection-type>
            <periodic>
              <idle-timeout>300</idle-timeout>
              <reconnect-timeout>60</reconnect-timeout>
            </periodic>
          </connection-type>
          <reconnect-strategy>
            <start-with>last-connected</start-with>
            <max-attempts>3</max-attempts>
          </reconnect-strategy>
        </netconf-client>
      </call-home>
    </netconf-server>
```

```
  </configuration>
```

```
</bootstrap-information>
```

9.3. YANG Module

The Zero Touch Information artifact is normatively defined by the YANG module defined in this section.

Note: the module defined herein uses data types defined in [RFC5280], [RFC6234], and [RFC6991].

```
<CODE BEGINS> file "ietf-zerotouch-information@2017-03-13.yang"

module ietf-zerotouch-information {
  yang-version "1.1";

  namespace "urn:ietf:params:xml:ns:yang:ietf-zerotouch-information";
  prefix    "zti";

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }

  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 Zero Touch Information
    artifact defined by RFC XXXX: Zero Touch Provisioning for NETCONF
    or RESTCONF based Management.

    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 "2017-03-13" {
     description
      "Initial version";
     reference
      "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
        Management";
   }

   rc:yang-data zerotouch-information {
     uses zerotouch-information-grouping;
   }

   grouping zerotouch-information-grouping {
     description
       "Defines the zerotouch information data model.  Grouping
        exists only to enable pyang tree output.";

     choice information-type {
       mandatory true;
       description
         "This choice statement ensures the response only contains
          redirect-information or bootstrap-information.  Note that
          this is the only mandatory true node, as the other nodes
          are not needed when the device trusts the bootstrap server,
          in which case the data does not need to be signed.";

       container redirect-information {
         description
           "This is redirect information, as described in Section 3.1
            in RFC XXXX.  Its purpose is to redirect a device to another
            bootstrap server.";
         reference
           "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF
            based Management";

         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.";
          }
          leaf trust-anchor { //should there be two fields like voucher?
            type binary;
            description
              "An X.509 v3 certificate structure as specified by RFC
               5280, Section 4, encoded using ASN.1 distinguished
               encoding rules (DER), as specified in ITU-T X.690.  A
               certificate that a device can use as a trust anchor
               to authenticate the bootstrap server it is being
               redirected to.";
            reference
              "RFC 5280:
                 Internet X.509 Public Key Infrastructure Certificate
                 and Certificate Revocation List (CRL) Profile.
               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).";
          }
        }
      }

      container bootstrap-information {

        description
          "This is bootstrap information, as described in Section 3.2 in
           RFC XXXX.  Its purpose is to provide the device everything it
           needs to bootstrap itself.";
        reference
          "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF
           based Management";

        container boot-image {
          description
            "Specifies criteria for the boot image the device MUST
```

```
            be running.";

         leaf name {  // maybe this should be a regex?
           type string;
           mandatory true;
           description
             "The name of a software image that the device MUST
              be running in order to process the remaining nodes.";
         }
         choice hash-algorithm {
            mandatory true;
            description
              "Identifies the hash algorithm used.";
            leaf sha256 {
               type string;
               description
                 "The hex-encoded SHA-256 hash over the boot
                  image file.  This is used by the device to
                  verify a downloaded boot image file.";
               reference
                 "RFC 6234: US Secure Hash Algorithms.";
            }
         }
         leaf-list uri {
           type inet:uri;
           min-elements 1;
           description
             "An ordered list of URIs to where the boot-image file MAY
              be obtained.  Deployments MUST know in which URI schemes
              (http, ftp, etc.) a device supports.  If a secure scheme
              (e.g., https) is provided, a device MAY establish a
              provisional connection to the server, by blindly
              accepting the server's credentials (e.g., its TLS
              certificate)";
         }
       }

       leaf configuration-handling {
         type enumeration {
           enum merge {
             description
              "Merge configuration into existing running configuration.";
           }
           enum replace {
             description
               "Replace existing running configuration with the passed
                configuration.";
           }
```

```
          }
          description
            "This enumeration indicates how the server should process
             the provided configuration.  When not specified, the device
             MAY determine how to process the configuration using other
             means (e.g., vendor-specific metadata).";
        }

        leaf pre-configuration-script {
          type script;
          description
            "A script that, when present, is executed before the
             configuration has been processed.";
        }

        anydata configuration {
          must "../configuration-handling";
          description
            "Any configuration data model known to the device.  It may
             contain manufacturer-specific and/or standards-based data
             models.";
        }

        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.  The contents of the
       script are considered specific to the vendor, product line,
       and/or model of the device.

       If a script is erroneously provided to a device that does not
       support the execution of scripts, the device SHOULD send a
       'script-warning' notification message, but otherwise continue
       processing the bootstrapping data as if the script had not
```

```
        been present.

        The script returns exit status code '0' on success and non-zero
        on error, with accompanying stderr/stdout for logging purposes.
        In the case of an error, the exit status code will specify what
        the device should do.

        If the exit status code is greater than zero, then the device
        should assume that the script had a soft error, which the
        script believes does not affect manageability.  If the device
        obtained the bootstrap information from a bootstrap server,
        it SHOULD send a 'script-warning' notification message.

        If the exit status code is less than zero, the device should
        assume the script had a hard error, which the script believes
        will affect manageability.  In this case, the device SHOULD
        send a 'script-error' notification message followed by a
        reset that will force a new boot-image install (wiping out
        anything the script may have done) and restart the entire
        bootstrapping process again.";
   }

 }

<CODE ENDS>
```

10. The Zero Touch Bootstrap Server API

This section defines a YANG [RFC6020] module that is used to define the RESTCONF [RFC8040] API used by the bootstrap server defined in Section 6.4. Examples illustrating this API in use are provided in Section 10.2.

10.1. Tree Diagram

The following tree diagram provides an overview for the bootstrap server RESTCONF API. The syntax used for this tree diagram is described in Section 1.4.

```
module: ietf-zerotouch-bootstrap-server
    +--ro device* [unique-id]
       +--ro unique-id                string
       +--ro zerotouch-information    pkcs7
       +--ro owner-certificate?       pkcs7
       +--ro ownership-voucher?       pkcs7
       +---x notification
          +---w input
             +---w notification-type    enumeration
             +---w message?             string
             +---w ssh-host-keys
             |  +---w ssh-host-key*
             |     +---w format      enumeration
             |     +---w key-data    string
             +---w trust-anchors
                +---w trust-anchor*
                   +---w protocol*      enumeration
                   +---w certificate    pkcs7
```

In the above diagram, notice that all of the protocol accessible nodes are read-only, to assert that devices can only pull data from the bootstrap server.

Also notice that the module defines an action statement, which devices use to provide progress notifications to the bootstrap server.

10.2. Example Usage

This section presents some examples illustrating the bootstrap server's API. Two examples are provided, one illustrating a device fetching bootstrapping data from the server, and the other illustrating a data posting a progress notification to the server.

The following example illustrates a device using the API to fetch its bootstrapping data from the bootstrap server. In this example, the device receives a signed response; an unsigned response would look similar except the last two fields (owner-certificate and ownership-voucher) would be absent in the response.

```
REQUEST
-------
['\' line wrapping added for formatting only]

GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server:\
device=123456 HTTP/1.1
HOST: example.com
Accept: application/yang.data+xml


RESPONSE
--------

HTTP/1.1 200 OK
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+xml

<!-- '\' line wrapping added for formatting purposes only -->

<device
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <unique-id>123456789</unique-id>
  <zerotouch-information>\
    Base64-encoded Zero Touch Information artifact\
  <zerotouch-information>
  <owner-certificate>Base64-encoded PKCS#7</owner-certificate>
  <ownership-voucher>Base64-encoded Voucher</ownership-voucher>
</device>
```

The following example illustrates a device using the API to post a notification to a bootstrap server. Illustrated below is the 'bootstrap-complete' message, but the device may send other notifications to the server while bootstrapping (e.g., to provide status updates). In this message, the device is sending both its SSH host keys and TLS server certificate, which the bootstrap server may, for example, pass to an NMS, as discussed in Section 7.3.

Note that devices that are able to present an IDevID certificate [Std-802.1AR-2009] when establishing SSH or TLS connections do not need to include its DevID certificate in the bootstrap-complete message. It is unnecessary to send the DevID certificate in this case because the IDevID certificate does not need to be pinned by an NMS in order to be trusted.

Note that the bootstrap server MUST NOT process a notification from a device without first authenticating the device. This is in contrast to when a device is fetching data from the server, a read-only

operation, in which case device authentication is not strictly required (e.g., when sending signed information).

```
REQUEST
-------
['\' line wrapping added for formatting only]

POST https://example.com/restconf/data/ietf-zerotouch:\
device=123456/notification HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<!-- '\' line wrapping added for formatting purposes only -->

<input
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <notification-type>bootstrap-complete</notification-type>
  <message>example message</message>
  <ssh-host-keys>
    <ssh-host-key>
      <format>ssh-rsa</format>
      <key-data>Base64-encoded SSH RSA Public Key</key-data>
    </ssh-host-key>
    <ssh-host-key>
      <format>ssh-dsa</format>
      <key-data>Base64-encoded SSH DSA Public Key</key-data>
    </ssh-host-key>
  </ssh-host-keys>
  <trust-anchors>
    <trust-anchor>
      <protocol>netconf-ssh</protocol>
      <protocol>netconf-tls</protocol>
      <protocol>restconf-tls</protocol>
      <protocol>netconf-ch-ssh</protocol>
      <protocol>netconf-ch-tls</protocol>
      <protocol>restconf-ch-tls</protocol>
      <certificate>Base64-encoded X.509</certificate>
    </trust-anchor>
  </trust-anchors>
</input>


RESPONSE
--------

HTTP/1.1 204 No Content
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
```

10.3. YANG Module

The bootstrap server's device-facing API is normatively defined by the YANG module defined in this section.

Note: the module defined herein uses data types defined in [RFC2315], [RFC5280], and [I-D.ietf-anima-voucher].

```
<CODE BEGINS> file "ietf-zerotouch-bootstrap-server@2017-03-13.yang"

module ietf-zerotouch-bootstrap-server {
  yang-version "1.1";

  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>
    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>";

  description
   "This module defines an interface for bootstrap servers, as defined
    by RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
    Management.

    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 "2017-03-13" {
    description
     "Initial version";
    reference
     "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
```

```
      Management";
  }

  // typedefs

  typedef pkcs7 {
    type binary;
    description
      "A PKCS #7 SignedData structure, as specified by Section 9.1
       in RFC 2315, encoded using ASN.1 distinguished encoding rules
       (DER), as specified in ITU-T X.690.";
     reference
       "RFC 2315:
          PKCS #7: Cryptographic Message Syntax Version 1.5.
        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).";
  }

  // protocol accessible node

  list device {
    key unique-id;
    config false;

    description
      "A device's record entry.  This is the only RESTCONF resource
       that a device will GET, as described in Section 8.2 in RFC XXXX.
       Getting just this top-level node provides a device with all the
       data it needs in a single request.";
    reference
      "RFC XXXX: Zero Touch Provisioning for NETCONF or
       RESTCONF based Management";

    leaf unique-id {
      type string;
      description
        "A unique identifier for the device (e.g., serial number).
         Each device accesses its bootstrapping record by its unique
         identifier.";
    }

    leaf zerotouch-information {
      type pkcs7;
      mandatory true;
      description
```

```
        "A 'zerotouch-information' artifact, as described in Section
         4.1 of RFC XXXX.  When conveyed over an untrusted transport, in
         order to be processed by a device, this PKCS#7 SignedData
         structure  MUST contain a 'signerInfo' structure, described
         in Section 9.1 of RFC 2315, containing a signature generated
         using the owner's private key.";
      reference
        "RFC XXXX: Zero Touch Provisioning for NETCONF or
           RESTCONF based Management.
         RFC 2315:
           PKCS #7: Cryptographic Message Syntax Version 1.5";
    }

    leaf owner-certificate {
      type pkcs7;
      must "../zerotouch-information" {
        description
          "An 'zerotouch-information' artifact must be present
           whenever an ownership certificate is presented.";
      }
      description
        "An unsigned PKCS #7 SignedData structure, as specified by
         Section 9.1 in RFC 2315, encoded using ASN.1 distinguished
         encoding rules (DER), as specified in ITU-T X.690.

         This structure MUST contain the owner certificate and all
         intermediate certificates leading up to at least the trust
         anchor certificate specified in the ownership voucher.
         Additionally, if needed by the device, this structure MAY
         also contain suitably fresh CRL and or OCSP Responses.

         X.509 certificates and CRLs are described in RFC 5280.
         OCSP Responses are described in RFC 6960.";
      reference
       "RFC 2315:
          PKCS #7: Cryptographic Message Syntax Version 1.5.
        RFC 5280:
          Internet X.509 Public Key Infrastructure Certificate
          and Certificate Revocation List (CRL) Profile.
        RFC 6960:
          X.509 Internet Public Key Infrastructure Online
          Certificate Status Protocol - OCSP.
        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).";
    }
```

```
    leaf ownership-voucher {
      type pkcs7;
      must "../owner-certificate" {
        description
          "An owner certificate must be present whenever an ownership
           voucher is presented.";
      }
      description
        "A 'voucher' artifact, as described in Section 5 of
         I-D.ietf-anima-voucher.  The voucher's 'device-identifier'
         MUST reference both the device's unique identifier and
         the owner's owner certificate.";
      reference
        "I-D.etf-anima-voucher:
           Voucher and Voucher Revocation Profiles for Bootstrapping
           Protocols";
    }

    action notification {
      input {
        leaf notification-type {
          type enumeration {
            enum bootstrap-initiated {
              description
                "Indicates that the device has just accessed the
                 bootstrap server.  The 'message' field below MAY
                 contain any additional information that the
                 manufacturer thinks might be useful.";
            }
            enum parsing-warning {
              description
                "Indicates that the device had a non-fatal error when
                 parsing the response from the bootstrap server.  The
                 'message' field 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, because the ownership voucher didn't
                 include the device's unique identifier, or because the
                 signature didn't match.  The 'message' field below
                 SHOULD indicate the specific error.  This notification
                 also indicates that the device has abandoned trying to
                 bootstrap off this bootstrap server.";
```

```
            }
            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' field
                 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' field SHOULD indicate the specific error
                 that occurred.  This notification also indicates
                 that the device has abandoned trying to bootstrap
                 off this bootstrap server.";
            }
            enum pre-script-warning {
              description
                "Indicates that the device obtained a greater than
                 zero exit status code from the script when it was
                 executed.  The 'message' field below SHOULD indicate
                 both the resulting exit status code, as well as
                 capture any stdout/stderr messages the script may
                 have produced.";
            }
            enum pre-script-error {
              description
                "Indicates that the device obtained a less than zero
                 exit status code from the script when it was executed.
                 The 'message' field below SHOULD indicate both the
                 resulting exit status code, as well as capture any
                 stdout/stderr messages the script may have produced.
                 This notification also indicates that the device has
                 abandoned trying to bootstrap off this bootstrap
                 server.";
            }
            enum config-warning {
              description
                "Indicates that the device obtained warning messages
                 when it committed the initial configuration.  The
                 'message' field 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' field below SHOULD indicate the error
                 messages that were generated.  This notification
                 also indicates that the device has abandoned trying
                 to bootstrap off this bootstrap server.";
            }
            enum post-script-warning {
              description
                "Indicates that the device obtained a greater than
                 zero exit status code from the script when it was
                 executed.  The 'message' field below SHOULD indicate
                 both the resulting exit status code, as well as
                 capture any stdout/stderr messages the script may
                 have produced.";
            }
            enum post-script-error {
              description
                "Indicates that the device obtained a less than zero
                 exit status code from the script when it was executed.
                 The 'message' field below SHOULD indicate both the
                 resulting exit status code, as well as capture any
                 stdout/stderr messages the script may have produced.
                 This notification also indicates that the device has
                 abandoned trying to bootstrap off this bootstrap
                 server.";
            }
            enum bootstrap-complete {
              description
                "Indicates that the device successfully processed the
                 all the bootstrapping data and that it is ready to be
                 managed.  The 'message' field below MAY contain any
                 additional information that the manufacturer thinks
                 might be useful.  After sending this notification,
                 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 notification-type. For instance, a
                 message indicating that the device is about to reboot
                 after having installed a boot-image could be provided.
                 The 'message' field below SHOULD contain information
                 that the manufacturer thinks might be useful.";
            }
```

```
          }
          mandatory true;
          description
            "The type of notification provided.";
        }
        leaf message {
          type string;
          description
            "An optional human-readable value.";
        }
        container ssh-host-keys {
          when "../notification-type = 'bootstrap-complete'" {
            description
              "SSH host keys are only sent when the notification
               type is 'bootstrap-complete'.";
          }
          description
            "A list of SSH host keys an NMS may use to authenticate
             a NETCONF connection to the device with.";
          list ssh-host-key {
            description
              "An SSH host-key";
            leaf format {
              type enumeration {
                enum ssh-dss { description "ssh-dss"; }
                enum ssh-rsa { description "ssh-rsa"; }
              }
              mandatory true;
              description
                "The format of the SSH host key.";
            }
            leaf key-data {
              type string;
              mandatory true;
              description
                "The key data for the SSH host key";
            }
          }
        }
        container trust-anchors {
          when "../notification-type = 'bootstrap-complete'" {
            description
              "Trust anchors are only sent when the notification
               type is 'bootstrap-complete'.";
          }
          description
            "A list of trust anchor certificates an NMS may use to
             authenticate a NETCONF or RESTCONF connection to the
```

```
             device with.";
          list trust-anchor {
            description
              "A list of trust anchor certificates an NMS may use to
               authenticate a NETCONF or RESTCONF connection to the
               device with.";
            leaf-list protocol {
              type enumeration {
                enum netconf-ssh     { description "netconf-ssh"; }
                enum netconf-tls     { description "netconf-tls"; }
                enum restconf-tls    { description "restconf-tls"; }
                enum netconf-ch-ssh  { description "netconf-ch-ssh"; }
                enum netconf-ch-tls  { description "netconf-ch-tls"; }
                enum restconf-ch-tls { description "restconf-ch-tls"; }
              }
              min-elements 1;
              description
                "The protocols that this trust anchor secures.";
            }
            leaf certificate {
              type pkcs7;
              mandatory true;
              description
                "An X.509 v3 certificate structure, as specified by
                 Section 4 in RFC5280, encoded using ASN.1 distinguished
                 encoding rules (DER), as specified in ITU-T X.690.";
              reference
                "RFC 5280:
                   Internet X.509 Public Key Infrastructure Certificate
                   and Certificate Revocation List (CRL) Profile.
                 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).";
            }
          }
        }
      }
    } // end action

  } // end device

}

<CODE ENDS>
```

11. Security Considerations

11.1. 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.

11.2. 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 MUST 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 and owner certificates are not revokable. In real-world terms, this means that manufacturers SHOULD only issue a single ownership voucher for the lifetime of some devices.

It is important to note that implementations SHOULD NOT rely on NTP for time, as it is not a secure protocol.

11.3. 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, do not send their IDevID certificate for client authentication, and they do not POST any progress notifications, and they assert that data downloaded from the server is signed.

11.4. 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 the notAfter value 99991231235959Z). Given the long-lived nature of these certificates, it is paramount to use a strong key length (e.g., 512-bit ECC).

11.5. 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.

11.6. 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.

12. IANA Considerations

12.1. The BOOTP Manufacturer Extensions and DHCP Options Registry

The following registrations are in accordance to RFC 2939 [RFC2939] for "BOOTP Manufacturer Extensions and DHCP Options" registry maintained at http://www.iana.org/assignments/bootp-dhcp-parameters.

Tag: XXX

Name: Zero Touch Information

Returns up to three zero touch bootstrapping artifacts.

```
 Code   Len
+-----+-----+----------------------+
| XXX |  n  | zerotouch-information |
+-----+-----+----------------------+

+-------------------+------------------+
| owner-certificate | ownership-voucher |
+-------------------+------------------+
```

Reference: RFC XXXX

Tag: YYY

Name: Zero Touch Information

Returns up to three zero touch bootstrapping artifacts.

```
 Code   Len
+-----+-----+----------------------+
| XXX |  n  | zerotouch-information |
+-----+-----+----------------------+

+-------------------+------------------+
| owner-certificate | ownership-voucher |
+-------------------+------------------+
```

Reference: RFC XXXX

12.2. The IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-zerotouch
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

12.3. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format defined in [RFC6020], the the following registrations are requested:

```
name:         ietf-zerotouch-information
namespace:    urn:ietf:params:xml:ns:yang:ietf-zerotouch
prefix:       zt
reference:    RFC XXXX

name:         ietf-zerotouch-bootstrap-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-zerotouch
prefix:       zt
reference:    RFC XXXX
```

13. Other Considerations

Both this document and [I-D.ietf-anima-bootstrapping-keyinfra] define bootstrapping mechanisms. The authors have collaborated on both solutions and believe that each solution has merit and, in fact, can work together. That is, it is possible for a device to support both solutions simultaneously.

14. Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): David Harrington, Michael Behringer, Dean Bogdanovic, Martin Bjorklund, Joe Clarke, Toerless Eckert, Stephen Farrell, Ian Farrer, Stephen Hanna, Wes Hardaker, Russ Mundy, Reinaldo Penno, Randy Presuhn, Max Pritikin, Michael Richardson, Phil Shafer, 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.

15. References

15.1. Normative References

[I-D.ietf-anima-voucher]
Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "Voucher and Voucher Revocation Profiles for Bootstrapping Protocols", draft-ietf-anima-voucher-00 (work in progress), January 2017.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <http://www.rfc-editor.org/info/rfc2315>.

[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, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://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, <http://www.rfc-editor.org/info/rfc6125>.

[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, <http://www.rfc-editor.org/info/rfc6234>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <http://www.rfc-editor.org/info/rfc6763>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <http://www.rfc-editor.org/info/rfc7468>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

[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>.

15.2. Informative References

[I-D.ietf-anima-bootstrapping-keyinfra] Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-04 (work in progress), October 2016.

[I-D.ietf-netconf-netconf-client-server] Watsen, K., Wu, G., and J. Schoenwaelder, "NETCONF Client and Server Models", draft-ietf-netconf-netconf-client-server-01 (work in progress), November 2016.

[RFC2939] Droms, R., "Procedures and IANA Guidelines for Definition of New DHCP Options and Message Types", BCP 43, RFC 2939, DOI 10.17487/RFC2939, September 2000, <http://www.rfc-editor.org/info/rfc2939>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[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, <http://www.rfc-editor.org/info/rfc6698>.

[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, <http://www.rfc-editor.org/info/rfc6960>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <http://www.rfc-editor.org/info/rfc7317>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <http://www.rfc-editor.org/info/rfc8071>.

Appendix A. Change Log

A.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 Zero Touch 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

A.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 Zero Touch Information DHCP Option
- o Added IANA request for media types for boot-image and configuration

A.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 manufacturer signed ownership vouchers and owner certificates.

- Renamed configuration server to bootstrap server, a more representative name given the information devices download from it.

- Replaced the concept of a configlet by defining a southbound interface for the bootstrap server using YANG.

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

A.4. 02 to 03

- Minor update, mostly just to add an Editor's Note to show how this draft might integrate with the draft-pritikin-anima-bootstrapping-keyinfra.

A.5. 03 to 04

- Major update formally introducing unsigned data and support for Internet-based redirect servers.

- Added many terms to Terminology section.

- Added all new "Guiding Principles" section.

- Added all new "Sources for Bootstrapping Data" section.

- Rewrote the "Interactions" section and renamed it "Workflow Overview".

A.6. 04 to 05

- Semi-major update, refactoring the document into more logical parts

- Created new section for information types

- Added support for DNS servers

- Now allows provisional TLS connections

- Bootstrapping data now supports scripts

- Device Details section overhauled

- Security Considerations expanded

- Filled in enumerations for notification types

A.7. 05 to 06

- Minor update
- Added many Normative and Informative references.
- Added new section Other Considerations.

A.8. 06 to 07

- Minor update
- Added an Editorial Note section for RFC Editor.
- Updated the IANA Considerations section.

A.9. 07 to 08

- Minor update
- Updated to reflect review from Michael Richardson.

A.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).

A.11. 09 to 10

- Factored ownership vocher 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.

A.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.

A.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's encoded, matching the language that was in Appendix B.

A.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.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Mikael Abrahamsson
T-Systems

EMail: "mikael.abrahamsson@t-systems.se