NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

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

October 19, 2015

Zero Touch Provisioning for NETCONF Call Home
draft-ietf-netconf-zerotouch-04

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 21, 2016.

Table of Contents

1. Introduction

A fundamental business requirement is to reduce costs where possible. For network operators, deploying devices to many locations can be a significant cost, as sending trained specialists to each site to do installations is both cost prohibitive and does not scale.

This document defines bootstrapping strategies enabling a device to securely obtain bootstrapping data with no installer input beyond racking the device and applying power. This bootstrapping data directs the device to install a boot image and an initial configuration, which enables the establishment of a NETCONF [RFC6241] or RESTCONF [draft-ietf-netconf-restconf] connection to its rightful owner's network management system (NMS).

In order to enable a NETCONF or RESTCONF connection to be established, the initial configuration should include settings such as enabling the NETCONF/RESTCONF service, including parameters needed to support an NMS-initiated or device-initiated connection, and configuring a local administrator account. Examples used in this draft illustrate this using models defined by [RFC7317] and [draft-ietf-netconf-server-model].

1.1. Use Cases

o Connecting to a remotely administered network

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

reach out to the public Internet for a well-known service it can bootstrap off of.

- 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 site-specific 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

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].

This document defines the following terms:

Artifact: The term "artifact" is used throughout to represent bootstrapping data that can be encoded outside of the RESTCONF protocol. For example, an artifact may be a file on disk or a message in another protocol. Unless used inside a secure protocol, artifacts must be signed and need to be provided along with an Owner Certificate and an Ownership Voucher (see terms), so the a device can validate the artifact's signature to its Rightful Owner (see term).

Bootstrap Server: The term "bootstrap server" is used within this document to mean any RESTCONF server implementing the YANG module defined in Section 6.4.

Device: The term "device" is used throughout this document to refer to the network element that needs to be bootstrapped. The device is the RESTCONF client to a Bootstrap Server (see above) and, at the end of bootstrapping process, the device is the NETCONF or RESTCONF server to a deployment-specific NMS. See Section 5 for more information about devices.

Network Management System (NMS): The acronym "NMS" is used throughout this document to refer to the deployment specific management system that the bootstrapping process ultimately connects the 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.

Owner Certificate: An owner certificate, signed by the device's manufacturer or delegate, binds an owner identity to the owner's private key, which the owner can subsequently use to sign artifacts. The owner certificate is an X.509 certificate encoding the owner's identity in the Subject field of the X.509 certificate. The owner certificate is used by devices only when validating owner signatures on Signed Data (see term).

Ownership Voucher: An ownership voucher, signed by the device's manufacturer or delegate, binds an owner identity to one or more device identities (e.g., serial numbers). The ownership voucher is used by devices only when validating owner signatures on Signed Data (see term).

Redirect Server: The term "redirect server" is used to refer to a Bootstrap Server (see above) that only returns Redirect Information (Section 2.4).

Rightful Owner: The rightful owner of a device is the person or organization that purchased the device. How ownership can be conveyed to a device is described in Section 2.3.

Secure Redirect: Secure redirect is like an HTTP Redirect except that it also returns TLS certificates that can be used as trust anchors to validate the secure connection to the Bootstrap Server the device is being redirected to.

Signed Data: The term "signed data" is used throughout to mean data that has been signed by a device's Rightful Owner's private key. Any time data is signed, it must be presented along with an Owner Certificate and Ownership Voucher (see terms).

Unsigned Data: The term "unsigned data" is used throughout to mean data that has not been signed by a device's Rightful Owner's private key. The option to use unsigned data is available only when the data is obtained over a secure connection, such as to a Redirect Server or a Bootstrap Server (see terms).

1.3. Tree Diagrams

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

o Brackets "[" and "]" enclose list keys.

- Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.

- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

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

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

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

2. Guiding Principles

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

2.1. Trust Anchors

A device in its factory default state can only trust remote keys for which it has preconfigured trust anchors. For instance, the device may have a trust anchor (e.g., a X.509 certificate) for when authenticating a very specific HTTPS server, and another trust anchor for when validating boot-image files, and yet another trust anchor for when verifying software licenses.

2.2. Conveying Trust

Trust can be conveyed by either transport level security or artifact signing. For instance, if a device connects to an HTTPS server, authenticating the TLS certificate to a known trust anchor, then any data the device receives from the HTTPS server can also be trusted. Likewise, if a device can authenticate the signature over some data to a known trust anchor, then that data can also be trusted. In general, any data obtained from a trusted source MAY be trusted and, any data obtained from an untrusted source MUST NOT be trusted.

It is possible but unnecessary to provide signed data over a secure connection. For instance, a device connecting to a trusted HTTPS server may retrieve data that has been signed by its rightful owner, but this is not required, as the device is already assured by the server that its data was staged by its rightful owner. That said, when an insecure connection is used (e.g., DHCP), the device has no choice but to require that the data be signed, in order to trust the data.

2.3. Ownership

The goal of this document is to enable a device to connect with its rightful owner's NMS. This entails the manufacturer being able to track who owns which devices (out of the scope of this document), as well as an ability to convey that information to devices (in scope). Matching the two ways to convey trust, this document provides both a protocol-oriented solution as well as an artifact based solution for conveying ownership.

The protocol based solution conveys ownership by API contract, in that the server asserts that it will only return data that it is sure was staged by that device's rightful owner. How ownership for a device is assured is out of scope of this document.

The artifact based solution involves the manufacturer signing an owner key and then later, when the ownership for devices is established, the manufacturer signing a voucher that assigns those devices to the owner, and then the owner using their private key to sign the artifacts. Thus, from the device's perspective, it can use the presented "ownership voucher" to validate the presented "owner certificate", which it can then use to validate the signature over the presented artifact.

The YANG module in Section 6.4 includes grouping statements defining the format for the owner certificates and ownership vouchers used by the bootstrapping solution presented in this document.

2.4. Information Types

This document presumes there exists two types of zero touch information: redirect information and bootstrapping information. Either type of data may by accessed as unsigned data over a secure connection to a trusted server (e.g., HTTPS), or as signed artifacts obtained via an insecure method (DHCP server, removable storage device, etc.).

The redirect information type of data provides two bits of information: bootstrap server locations and trust anchors. The trust anchors are provided to enable the device to authentic the specified bootstrap servers (TLS certificate-based authentication). This is what distinguishes this technique from a standard HTTP Redirect and why it may sometimes be called "secure redirect".

The bootstrap information type of data provides information describing the boot-image and configuration the device should be running, in order to be considered bootstrapped. The boot-image

information is optional but, if it is provided, the device should install the boot image prior to installing the configuration.

The YANG module in Section 6.4 includes grouping statements defining the format for redirect and bootstrap information types used by the bootstrapping solution presented in this document.

3. Sources for Bootstrapping Data

Following are the sources of bootstrapping data that are referenced by the workflow presented in Section 4.3. Other sources for bootstrapping information may be described in other documents, so long as the principles for when the bootstrapping data needs to be signed or not are enforced.

Each of the descriptions below show how the bootstrapping data needs to be handled in a manner consistent with the guiding principles in Section 2.

For devices supporting more than one source for bootstrapping data, no particular sequencing order has to be observed, as each source is equally secure, in that the chain of trust always goes back to the same root of trust, the manufacturer.

3.1. Removable Storage

A device may attempt to read bootstrapping information from a directly attached removable storage device. This information would most likely have to be signed, as removable storage devices are generally not trustworthy.

The information loaded from a removable storage device may redirect the device to a bootstrap server (i.e., redirect information) or it may provide the boot image and configuration (i.e., bootstrapping information) directly. For when providing the information directly, even the raw boot image file could be on the removable storage device, making it a fully self-standing solution.

3.2. DHCP Server

A device may attempt to read bootstrapping information from a DHCP server (e.g., DHCP options). This information would have to be signed, as the DHCP protocol is not a secure protocol.

The information may again be either redirect or bootstrapping information. If bootstrapping information is provided, the URI to the boot image would have to specify a file server (e.g., ftp://, tftp://, etc.), as DHCP servers do not themselves distribute files.

Note that it is acceptable for boot images to be fetched using an insecure protocol when having an embedded signature, as is commonly the case.

3.3. Internet Based Service

A device may attempt to read bootstrapping information from a trusted Internet based service. The hosted information would not have to be signed, as the device would authenticate the service when establishing a secure connection to it, using trust anchors the device is manufactured with in its factory default state.

This document defines a RESTCONF API for a bootstrap server that may be hosted on the Internet. The YANG module describing this API is provided in Section 6.4.

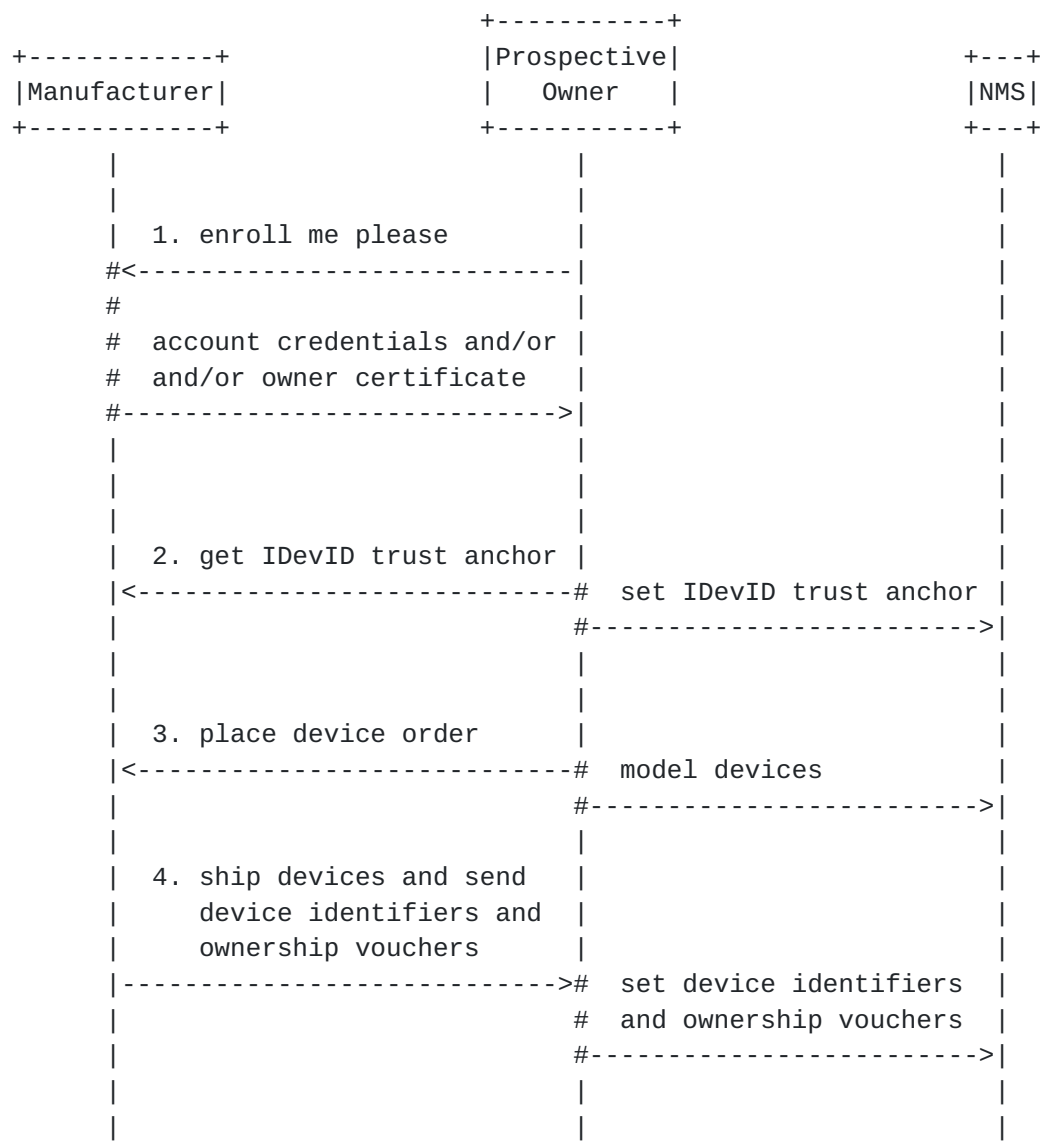
The information may again redirect the device to a bootstrap server (i.e., redirect information) or it may direct the device to load a boot image and a configuration (i.e., bootstrapping information). If bootstrapping information is provided, the URI to the boot image would not have to be to a server the device has a trust anchor for, assuming the boot image has an embedded signature, as is commonly the case.

4. 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.

4.1. Onboarding and Ordering Devices

The following diagram illustrates key interactions that occur from when a manufacturer or delegate onboards a prospective device owner to when the manufacturer ships devices for an order placed by the prospective device owner.



The interactions in the above diagram are described below.

1. A prospective owner establishes a trust relationship with a manufacturer in order to place zero touch orders. Assuming the manufacturer or delegate hosts a secure redirect server, this onbording interaction might entail the creation of an online account that the owner can use to configure redirect information for future device orders. Alternatively, the onbording interaction may include the manufacturer signing an owner certificate (see Section 1.2), to be used for bootstrapping devices not using the manufacturer's redirect server. The onboarding interaction may also do both, giving the choice to the owner for how specific devices should bootstrap.

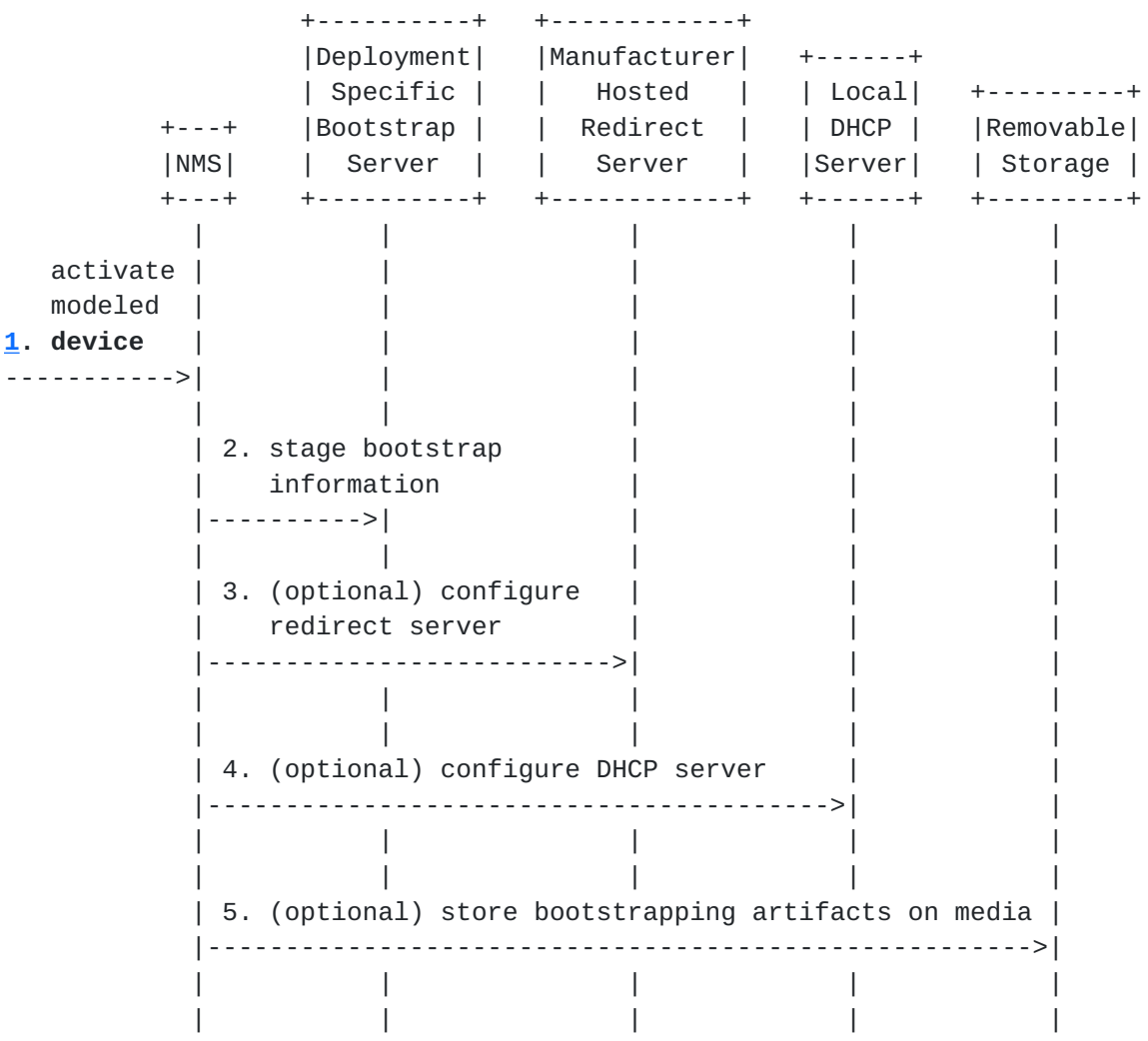
2. The prospective owner downloads from the manufacturer the X.509 based trust anchor certificate that can be used to validate the IDevID certificate [Std-802.1AR-2009] the devices will present as their SSH host key or TLS server certificate, when establishing a NETCONF or RESTCONF connection with the prospective owner's deployment-specific NMS.

3. Some time later, the prospective owner places an order with the manufacturer, perhaps with a special flag checked for zero touch handling. At this time, perhaps before placing the order, the owner may model the devices in their NMS. That is, create virtual objects for the devices with no real-world device associations. For instance the model can be used to simulate the device's location in the network and the configuration it should have when fully operational.

4. When the manufacturer ships the devices for the order, the manufacturer notifies the owner of the devices' unique identifiers and shipping destinations, which the owner can use to stage the network for when the devices powers on. Additionally, the manufacturer may send a ownership voucher assigning ownership of those devices to the rightful owner and/or configure backend systems so the secure redirect service can associate the redirect information to the devices. The owner sets this information on the NMS, perhaps binding specific device identifiers and ownership vouchers (if supported) to specific modeled devices.

4.2. Owner Stages the Network for Bootstrap

The following diagram illustrates how an owner stages the network for bootstrapping devices.



The interactions in the above diagram are described below.

1. Having previously modeled the devices, including setting their fully operational configurations, associating device identifiers and ownership vouchers (if supported), the owner may "activate" one or more modeled devices. That is, tell the NMS to perform the steps necessary to prepare for when the real-world devices are powered up.

2. One thing the NMS must do is configure the deployment specific bootstrap server. Illustrated here as an external component, the bootstrap server may be implemented as an internal component of the NMS itself. Configuring the bootstrap server may occur via a programmatic API not defined by this document. This step sets signed or unsigned bootstrap information, as shown in Section 6.2, for the devices being activated. The configuration set MUST be at least enough to enable a secure NETCONF or

RESTCONF connection to be established and MAY be the device's fully operational configuration.

3. If it is desired to use a manufacturer or delegate hosted redirect service to supply the bootstrapping information, the redirect server would need to be configured to supply the redirect information to the devices. Configuring the redirect server may occur via a programmatic API not defined by this document. This step sets signed or unsigned redirect information, as shown in Section 6.2, for the devices being activated. The redirect information MUST set the IP address or hostname of the deployment specific bootstrap server and MAY set the X.509 trust anchor certificate to authenticate the bootstrap server's TLS certificate.

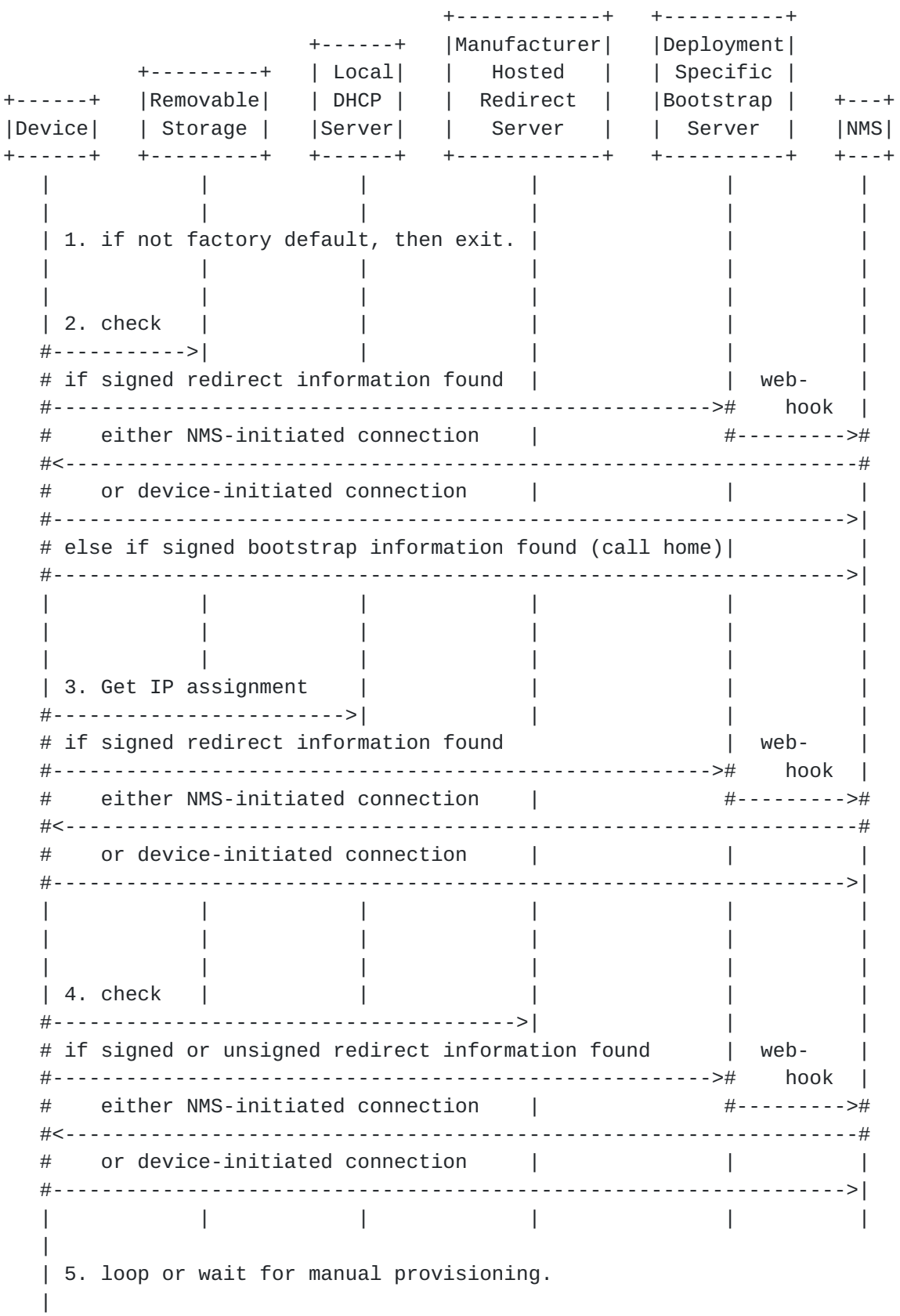
4. If it is desired to use a DHCP server to supply bootstrapping information, the DHCP server would need to be configured to supply the redirect information to the devices. Configuring the DHCP server may occur via a programmatic API (not defined by this document). Since DHCP is an insecure protocol, the information would have to be signed. That is, either signed redirect or signed bootstrap information, as shown in Section 6.2.

5. 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. Since a removable storage device is insecure, the information would have to be signed. That is, either signed redirect or signed bootstrap information, as shown in Section 6.2.

4.3. Device Powers On

The following diagram illustrates how a device might behave when powered on. Note that this is merely exemplary, subject to which bootstrapping strategies the device supports, which may be more or less than depicted below.

This example sequences the sources of information (see Section 3) based on locality, or how "close" to the device the data is. Whether this sequence makes sense for a specific type of device needs to be determined by the manufacturer.


+------------+ +----------+
+------+ +---------+ +------+ |Manufacturer| |Deployment|
|Device| |Removable| | Local| | Hosted | | Specific |
| | | Storage | | DHCP | | Redirect | |Bootstrap | +---+
| | | | |Server| | Server | | Server | |NMS|
+------+ +---------+ +------+ +------------+ +----------+ +---+
| 1. if not factory default, then exit.
| 2. check
if signed redirect information found
either NMS-initiated connection
or device-initiated connection
else if signed bootstrap information found (call home)
| 3. Get IP assignment
if signed redirect information found
either NMS-initiated connection
or device-initiated connection
| 4. check
if signed or unsigned redirect information found
either NMS-initiated connection
or device-initiated connection
| 5. loop or wait for manual provisioning.
web-hook

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 has a modified state, then the bootstrapping logic would exit and none to the following interactions would occur.

2. If the device is able to load bootstrapping data from a removable storage device (e.g., USB flash drive), it might choose to do so first. The removable storage may have either signed redirect information or signed bootstrap information, as shown in Section 6.3.

   In the case that signed redirect information is found, the device would use it to established a connection to the deployment-specific bootstrap server, which would set its boot image and configure it to enable connections with the deployment-specific NMS to be established. If the bootstrap supports notifying (e.g., via a web-hook) external systems when a device sends its bootstrap-complete notification (Section 6.4), it would be possible for the NMS to initiate a NETCONF or RESTCONF connection to the device. Otherwise the configuration could configure the device it initiate a NETCONF or RESTCONF call home [draft-ietf-netconf-call-home] connection to the deployment-specific NMS.

   In the case that signed bootstrap information is found, the device would use it to set its boot image and initial configuration, which would have to direct it to initiate a NETCONF or RESTCONF call home connection to the deployment-specific NMS.

   If the device is unable to bootstrap using any of the information on the removable storage device, it would proceed to the next source of bootstrapping information, if any.

3. If the device is able to load bootstrapping data from a DHCP server, when obtaining a DHCP assignment, it may receive signed redirect information in a DHCP Option (Section 8). The device would process the signed redirect information in the same manner as described above for when it's loaded from a removable storage device. If the device is unable to bootstrap using information provided by a DHCP server, it would proceed to the next source of bootstrapping information, if any.

4. If the device is able to obtain a routable address to the Internet, it may attempt to establish a connection to a redirect server that is set by its factory default state (Section 5.1).

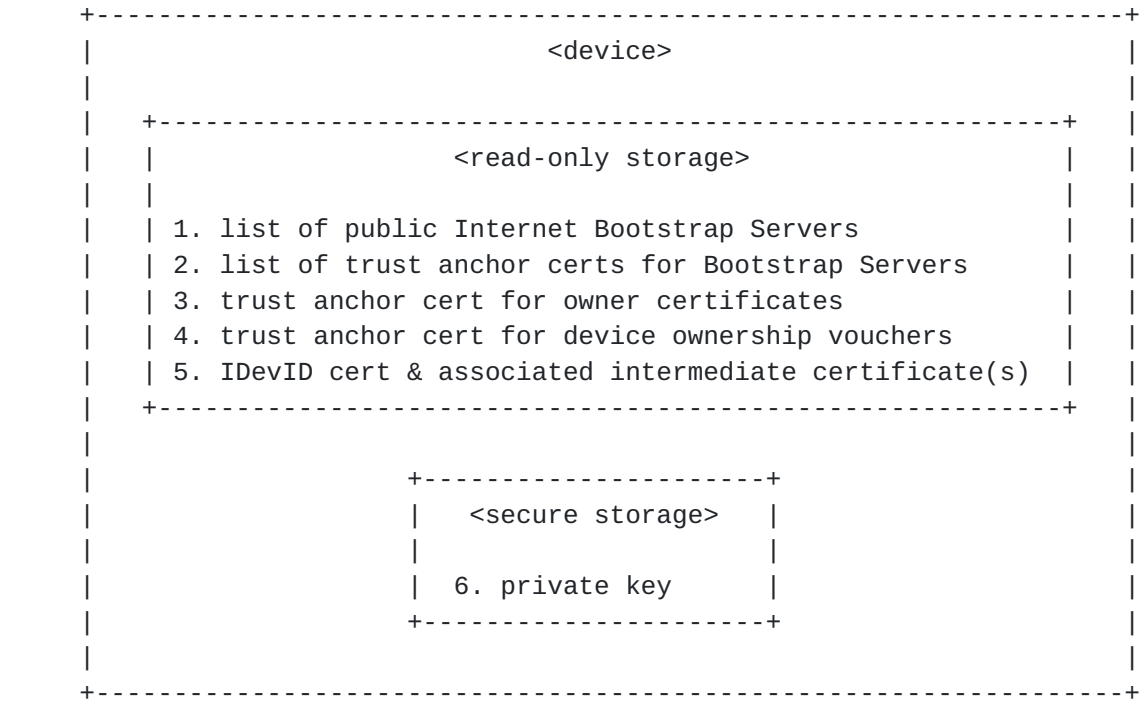
These connections would use the RESTCONF API described in this document and would be secured using trust anchors also set in the device's factory default state. The redirect server may provide signed or unsigned redirect information. In either case, the device would process the redirect information in the same manner as described above for when it's loaded from a removable storage device. If the device is unable to bootstrap using information provided by any redirect servers, it would proceed to the next source of bootstrapping information, if any.

5. If no more sources of bootstrapping information are available, the device may fall into a loop to try again or it may provide manageability interfaces for manual configuration (e.g., CLI, HTTP, NETCONF, etc.).

5. Device Details

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

5.1. Factory Default State



1. Devices that support loading bootstrapping information from the Internet (see Section 3) MUST be manufactured with a list of default Bootstrap Servers. Each Bootstrap Server may be identified via a hostname or an IP address.

2. Devices that support loading bootstrapping information from the Internet (see Section 3) SHOULD be manufactured with a list of trust anchor certificates that can be used to authenticate the Bootstrap Server connections with.

3. Devices that support loading owner signed data (see Section 1.2) MUST be manufactured with the trust anchor certificate for the Owner Certificates that the Manufacturer provides to prospective owners when they enroll in the Manufacturer's Zero Touch program (see Section 4.1).

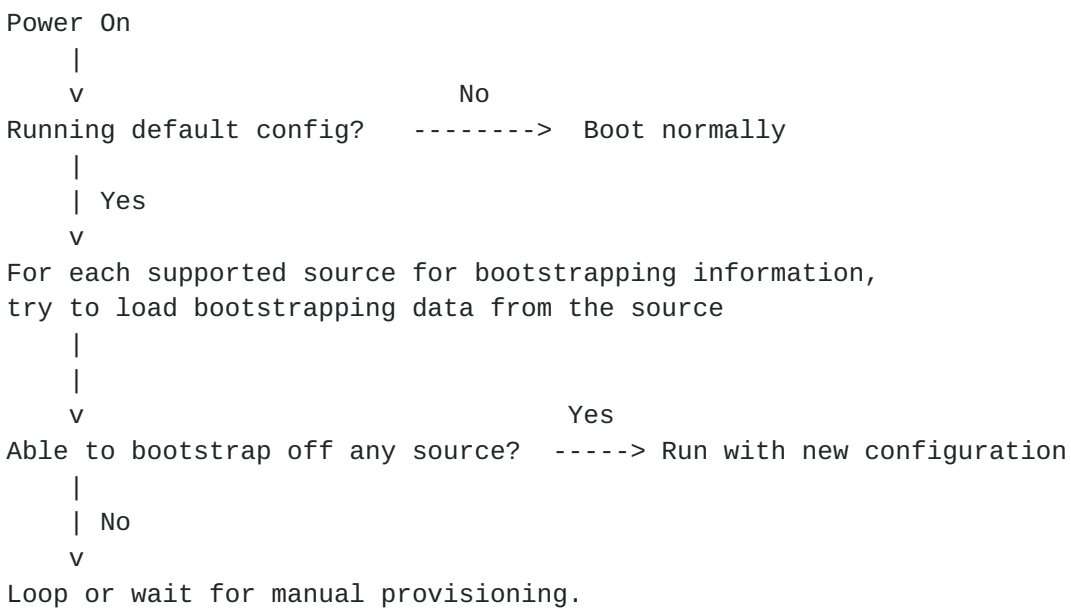
4. Devices that support loading owner signed data (see Section 1.2) MUST also be manufactured with the trust anchor certificate for the device Ownership Vouchers that the Manufacturer provides to prospective owners when it ships out an order of Zero Touch devices (see Section 4.1).

5. Devices MUST be manufactured with an initial device identifier (IDevID), as defined in [Std-802.1AR-2009]. The IDevID is an X.509 certificate, encoding a globally unique device identifier (e.g., serial number). The device MUST also possess any intermediate certificates between the IDevID certificate and the Manufacturer's IDevID trust anchor certificate.

6. 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).

5.2. Boot Sequence

A device claiming to support Zero Touch MUST support the boot sequence described in this section.



These interactions are described next.

1. When the device powers on, it first checks to see if it is running the factory default configuration. If it is running a modified configuration, then it boots normally.

2. The device iterates over its list of sources for bootstrapping information. Details for handling different types of sources are provided in subsequent sections.

3. If the device is able to bootstrap itself off any of the sources for bootstrapping information, it runs with the new bootstrapped configuration merged into its running datastore.

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

When the source is a removable storage device, the device MUST be able to read from it signed data (see term) and validate that the data was signed by its rightful owner, using the algorithm in Section 5.3.

When the source is a DHCP server, the device MUST be able to read from it signed data (see term) and validate that the data was signed by its rightful owner, using the algorithm in Section 5.3.

When the source is a bootstrap server, that is, using the RESTCONF API presented in Section 6.4, the device MUST be able to authenticate the server using one of the the device's preconfigured trust anchors.

Once done authenticating the bootstrap server, the device MUST attempt to fetch the bootstrapping data hosted for it there, using its unique identifier (e.g., serial number) as the key into the "device" list. If bootstrapping data is found and it is signed, then the device MUST first validate that the data was signed by its rightful owner using the algorithm in Section 5.3. The device then processes the bootstrapping data as described in Section 5.4. The device MAY post progress notification messages to the server, but SHOULD only do so if it has first authenticated itself to the server (e.g., client authentication).

5.3. Validating Signed Data

If the device is ever presented signed data, it MUST validate the signed data as described in this section.

Whenever there is signed data, the device MUST also be provided an Ownership Voucher and an Owner Certificate.

The device MUST first authenticate the Ownership Voucher by validating the signature on it to one of its preconfigured trust anchors (see Section 5.1) and verify that the voucher contains the device's unique identifier (e.g., serial number). If the authentication of the voucher is successful, the device extracts the Rightful Owner's identity from the voucher for use in the next step.

Next the device MUST authenticate the Owner Certificate by performing X.509 certificate path validation on it to one of its preconfigured trust anchors (see Section 5.1) and by verifying that the Subject contained in the certificate matches the Rightful Owner identity extracted from the voucher in the previous step. 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 authenticate the signed data by verifying the signature on it 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.

5.4. Processing Bootstrap Data

In order to process bootstrapping data, the device MUST follow the steps presented in this section.

If the data is redirect-information (see Section 2.4), the device MUST immediately attempt to establish a RESTCONF connection to the

provided bootstrap server IP address or hostname. If a hostname is provided and DNS resolves it to more than one IP address, the device MUST attempt to try to connect to all of them, until it is able to successfully bootstrap off one of them. The device MUST authenticate the bootstrap server's TLS certificate using the X.509 certificate provided by the redirect information.

If the data is bootstrap-information (see Section 2.4), the device MUST first check if it contains any boot-image information and, if so, check to see if it differs from what the device is currently running and, if so, install the boot-image using the provided URI and reboot (Note, it is assumed that the boot-image contains an embedded signature that the installation step will verify). This will cause the device's bootstrap logic to restart, which will again come to this point, though with a matching boot-image, thus letting the device to proceed past this step. Next the device MUST process the configuration contained in the bootstrapping information, by merging it into its running configuration.

At this point, the device has completely processed the bootstrapping data and is "bootstrap complete". If the configuration configured the device it initiate a call home connection, it would proceed to do so now. Otherwise, the device would wait for a NETCONF or RESTCONF client to connect to it.

6. YANG-defined API and Artifacts

Central to the solution presented in this document is the use of a YANG module [RFC6020] to simultaneously define a RESTCONF based API for a bootstrap/redirect server as well as the encoding for signed artifacts that can be conveyed outside of the RESTCONF protocol (DHCP, FTP, TFTP, etc.).

6.1. Module Overview

The following tree diagram Section 1.3 provides an overview for both the API and artifacts that can be used outside of RESTCONF.

```
module: ietf-zerotouch-bootstrap-server
   +--ro devices
      +--ro device* [unique-id]
         +--ro unique-id               string
         +--ro (type)?
         |  +--:(redirect-information)
         |  |  +--ro redirect-information
         |  |     +--ro address         inet:host
         |  |     +--ro trust-anchor    binary
         |  |     +--ro signature?      string
         |  +--:(bootstrap-information)
         |     +--ro bootstrap-information
         |        +--ro boot-image
         |        |  +--ro name         string
         |        |  +--ro md5          string
         |        |  +--ro sha1         string
         |        |  +--ro path         string
         |        |  +--ro signature?   string
         |        +--ro configuration
         |           +--ro config
         |           +--ro signature?   string
         +--ro ownership-voucher
         |  +--ro voucher       binary
         |  +--ro issuer-crl?   string
         +--ro owner-certificate
         |  +--ro certificate    string
         |  +--ro issuer-crl?    string
         +---x notification
            +---w input
               +---w type             enumeration
               +---w message?         string
               +---w ssh-host-keys
                  +---w ssh-host-key*
                     +---w format      enumeration
                     +---w key-data    string
```

In the above diagram, notice that all of the protocol accessible node 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 may use to provide progress notifications to the Bootstrap Server.

6.2. API Examples

This section presents some examples illustrating device interactions with a Bootstrap Server to access Redirect and Bootstrap information, both unsigned and signed, as well as to send a progress notification.

6.2.1. Unsigned Redirect Information

The following example illustrates a device using the API to fetch its bootstrapping data. In this example, the device receives unsigned redirect information. This example is representative of a response a well-known Internet facing redirect service might return.

```
REQUEST
-------
GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server::devices/
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

<devices xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <device>
    <unique-id>123456789</unique-id>
    <redirect-information>
      <address>phs.example.com</address>
      <trust-anchor>
        WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc3OEgrRkYyRTFwdSt4ZVRJbVFFM
        lLQllsdWpOcjFTMnRLR05EMUc2OVJpK2FWNGw2NTdZNCtadVJMZgpRYjk
        zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot
        NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPCk1CMEdBMVVkRGd
        VEJiZ0JTWEdlbUEKMnhpRHVOTVkvVHFLNWd4cFJBZ1ZOYUU0cERZd05ER
        V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNSll6UG8zREF
        NQmdOVkhSTUJBZjhFCkFqQUFNQTRHQTFVZER3RUIvd1FFQXdJSGdEQnBC
        Z05WSFI4RVlqQmdNRjZnSXFBZ2hoNW9kSFJ3T2k4dlpYaGgKYlhCc1pTN
        WpiMjB2WlhoaGJYQnNaUzVqY215aU9LUTJNRFF4Q3pBSkJnTlZCQVlUQW
        QmdOVkJBWVRBbFZUTVJBd0RnWURWUVFLRXdkbAplR0Z0Y0d4bE1RNHdEQ
        MkF6a3hqUDlVQWtHR0dvS1U1eUc1SVR0Wm0vK3B0R2FieXVDMjBRd2kvZ
        25PZnpZNEhONApXY0pTaUpZK2xtYWs3RTRORUZXZS9RdGp4NUlXZmdvN2
        RJSUJQFRStS0Cg==
      </trust-anchor>
    </redirect-information>
  </device>
</devices>
```

6.2.2. Signed Redirect Information

The following example illustrates a device using the API to fetch its bootstrapping data. In this example, the device receives signed redirect information. This example is representative of a response that redirect service might return if concerned the device might not be able to authenticate its TLS certificate.

```
REQUEST
```

```
GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server::devices/
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


<devices xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <device>
    <unique-id>123456789</unique-id>
    <redirect-information>
      <address>phs.example.com</address>
      <trust-anchor>
        WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc3OEgrRkYyRTFwdSt4ZVRJbVFFM
        lLQllsdWpOcjFTMnRLR05EMUc2OVJpK2FWNGw2NTdZNCtadVJMZgpRYjk
        zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot
        NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPCk1CMEdBMVVkRGd
        VEJiZ0JTWEdlbUEKMnhpRHVOTVkvVHFLNWd4cFJBZ1ZOYUU0cERZd05ER
        V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNSll6UG8zREF
        NQmdOVkhSTUJBZjhFCkFqQUFNQTRHQTFVZER3RUIvd1FFQXdJSGdEQnBC
        Z05WSFI4RVlqQmdNRjZnSXFBZ2hoNW9kSFJ3T2k4dlpYaGgKYlhCc1pTN
        WpiMjB2WlhoaGJYQnNaUzVqY215aU9LUTJNRFF4Q3pBSkJnTlZCQVlUQW
        QmdOVkJBWVRBbFZUTVJBd0RnWURWUVFLRXdkbAplR0Z0Y0d4bE1RNHdEQ
        MkF6a3hqUDlVQWtHR0dvS1U1eUc1SVR0Wm0vK3B0R2FieXVDMjBRd2kvZ
        25PZnpZNEhONApXY0pTaUpZK2xtYWs3RTRORUZXZS9RdGp4NUlXZmdvN2
        RJSUJQFRStS0Cg==
      </trust-anchor>
      <signature>
        SomeSignatureString
      </signature>
    </redirect-information>
    <ownership-voucher>
      <voucher>
        ChQQSnVuaXBlcl9OZXR3b3JrczEdMBsGA1UECxQUQ2VydGlmaWNhdGVfSXNzdWFu
        Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
        GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
```

X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v

```
        MjAO
      </voucher>
      <issuer-crl>
        QGp1bmlwZXIuY29tMB4XDTE0MDIyNzE0MTM1MloXDTE1MDIyNzE0MTM1MlowMDET
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        RDEuRiZNRNLeJpgN9YWkXLAZX2rASwy041EMmZ6KAkWUd3ZmXucfoLpdRemfuPii
        KQTpIM/rNrbrkuTmalezFoFS7mrxLXJAsfP1guVcD7sLCyjvegL8pRCCrU9xyKLF
        8u/Qz4s0x0uzcGYh0sd3iWj21+AtigSLdMD76/j/VzftQL8B1yp3vc1EZiowOwq4
        AwEAAaOCAW0wggFpMBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHppoyXF
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        NTOufhQsD2t4TYpEkzLEiZqSswdBOaPxPcJLQNW8Bw2xN+A9GX=
      </issuer-crl>
    </ownership-voucher>
    <owner-certificate>
      <certificate>
        MIIExTCCA62gAwIBAgIBATANBgkqhkiG9w0BAQsFADCBqjELMAkGA1UEBhMCVVMx
        EzARBgNVBAgTCkNhbGlmb3JuaWExEjAQBgNVBAcTCVN1bm55dmFsZTEZMBcGA1UE
        ChQQSnVuaXBlcl9OZXR3b3JrczEdMBsGA1UECxQUQ2VydGlmaWNhdGVfSXNzdWFu
        Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
        QGp1bmlwZXIuY29tMB4XDTE0MDIyNzE0MTM1MloXDTE1MDIyNzE0MTM1MlowMDET
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        RDEuRiZNRNLeJpgN9YWkXLAZX2rASwy041EMmZ6KAkWUd3ZmXucfoLpdRemfuPii
        ap1DgmS3IaYl/s4OOF8yzcYJprm8O7NyZp+Y9H1U/7Qfp97/KbqwCgkHSzOlnt0X
        KQTpIM/rNrbrkuTmalezFoFS7mrxLXJAsfP1guVcD7sLCyjvegL8pRCCrU9xyKLF
        8u/Qz4s0x0uzcGYh0sd3iWj21+AtigSLdMD76/j/VzftQL8B1yp3vc1EZiowOwq4
        KmORbiKU2GTGZkaCgCjmrWpvrYWLoXv/sf2nPLyK6YjiWsslOJtRO+KzRbs2B18C
        AwEAAaOCAW0wggFpMBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHppoyXF
        yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
        GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
        X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
        MjAOBgNVHQ8BAf8EBAMCAgQwQgYDVR0fBDswOTA3oDWgM4YxaHR0cDovL2NybC5q
        dW5pcGVyLm5ldD9jYT1KdW5pcGVyX1RydXN0X0FuY2hvcl9DQTANBgkqhkiG9w0B
        AQsFAAOCAQEAOuD7EBilqQcT3t2C4AXta1gGNNwdldLLw0jtk4BMiA9l//DZfskB
        2AaJtiseLTXsMF6MQwDs1YKkiXKLu7gBZDlJ6NiDwy1UnXhi2BDG+MYXQrc6p76K
        z3bsVwZlaJQCdF5sbggc1MyrsOu9QirnRZkIv3R8ndJH5K792ztLquulAcMfnK1Y
        NTOufhQsD2t4TYpEkzLEiZqSswdBOaPxPcJLQNW8Bw2xN+A9GX7WJzEbT/G7MUfo
        Sb+U2PVsQTDWEzUjVnG7vNWYxirnAOZ0OXEWWYxHUJntx6DsbXYuX7D1PkkNr7ir
        96DpOPtX7h8pxxGSDPBXIyvg02aFMphstQ==
      </certificate>
      <issuer-crl>
        Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
```

```
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
        GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
        X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
        MjAO==
      </issuer-crl>
    </owner-certificate>
  </device>
</devices>
```

6.2.3. Unsigned Bootstrap Information

The following example illustrates a device using the API to fetch its bootstrapping data. In this example, the device receives unsigned bootstrapping information. This example is representative of a response a locally deployed bootstrap server might return.

```
REQUEST
-------
GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server::devices/
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


<devices xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <device>
    <unique-id>123456789</unique-id>
    <bootstrap-information>
      <boot-image>
        <name>
          boot-image-v3.2R1.6.img
        </name>
        <md5>
          SomeMD5String
        </md5>
        <sha1>
          SomeSha1String
        </sha1>
        <path>
```

```
/some/path/to/raw/file
```

```
        </path>
      </boot-image>
      <configuration>
        <config>
          <!-- from ietf-system.yang -->
          <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
            <authentication>
              <user>
                <name>admin</name>
                <ssh-key>
                  <name>admin's rsa ssh host-key</name>
                  <algorithm>ssh-rsa</algorithm>
                  <key-data>AAAAB3NzaC1yc2EAAAADAQABAAABAQDeJMV8zrtsi8CgEsRC
                  jCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/O2Mwj
                  E1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcC
                  WAw1lOr9IDGDAuww6G45gLcHalHMmBtQxKnZdzU9kx/fL3ZS5G76Fy6sA5
                  vg7SLqQFPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2DKqBIe340jWq
                  EIuA7LvEJYql4unq4Iog+/+CiumTkmQIWRgIoj4FCzYkO9NvRE6fOSLLf6
                  gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</key-data>
                </ssh-key>
              </user>
            </authentication>
          </system>
          <!-- from ietf-netconf-server.yang -->
          <netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-
server">
            <call-home>
              <application>
                <name>config-mgr</name>
                <ssh>
                  <endpoints>
                    <endpoint>
                      <name>east-data-center</name>
                      <address>11.22.33.44</address>
                    </endpoint>
                    <endpoint>
                      <name>west-data-center</name>
                      <address>55.66.77.88</address>
                    </endpoint>
                  </endpoints>
                  <host-keys>
                    <host-key>my-call-home-x509-key</host-key>
                  </host-keys>
                </ssh>
              </application>
            </call-home>
          </netconf-server>
        </config>
```

```
      </configuration>
```

```
    </bootstrap-information>
  </device>
</devices>
```

6.2.4. Signed Bootstrap Information

The following example illustrates a device using the API to fetch its bootstrapping data. In this example, the device receives signed bootstrapping information. This example is representative of a response that bootstrap service might return if concerned the device might not be able to authenticate its TLS certificate.

```
REQUEST
-------
GET https://example.com/restconf/data/ietf-zerotouch-bootstrap-server::devices/
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

<devices xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <device>
    <unique-id>123456789</unique-id>
    <bootstrap-information>
      <boot-image>
        <name>
          boot-image-v3.2R1.6.img
        </name>
        <md5>
          SomeMD5String
        </md5>
        <sha1>
          SomeSha1String
        </sha1>
        <path>
          /some/path/to/raw/file
        </path>
        <signature>
          SomeSignatureString
        </signature>
      </boot-image>
```

```
      <configuration>
```

```
        <config>
          <!-- from ietf-system.yang -->
          <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
            <authentication>
              <user>
                <name>admin</name>
                <ssh-key>
                  <name>admin's rsa ssh host-key</name>
                  <algorithm>ssh-rsa</algorithm>
                  <key-data>AAAAB3NzaC1yc2EAAAADAQABAAABAQDeJMV8zrtsi8CgEsRC
                  jCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/O2Mwj
                  E1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcC
                  WAw1lOr9IDGDAuww6G45gLcHalHMmBtQxKnZdzU9kx/fL3ZS5G76Fy6sA5
                  vg7SLqQFPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2DKqBIe340jWq
                  EIuA7LvEJYql4unq4Iog+/+CiumTkmQIWRgIoj4FCzYkO9NvRE6fOSLLf6
                  gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</key-data>
                </ssh-key>
              </user>
            </authentication>
          </system>
          <!-- from ietf-netconf-server.yang -->
          <netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-
server">
            <call-home>
              <application>
                <name>config-mgr</name>
                <ssh>
                  <endpoints>
                    <endpoint>
                      <name>east-data-center</name>
                      <address>11.22.33.44</address>
                    </endpoint>
                    <endpoint>
                      <name>west-data-center</name>
                      <address>55.66.77.88</address>
                    </endpoint>
                  </endpoints>
                  <host-keys>
                    <host-key>my-call-home-x509-key</host-key>
                  </host-keys>
                </ssh>
              </application>
            </call-home>
          </netconf-server>
        </config>
        <signature>
          SomeSignatureString
        </signature>
```

```
      </configuration>
```

```
    </bootstrap-information>
    <ownership-voucher>
      <voucher>
        ChQQSnVuaXBlcl9OZXR3b3JrczEdMBsGA1UECxQUQ2VydGlmaWNhdGVfSXNzdWFu
        Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
        GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
        X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
        MjAO
      </voucher>
      <issuer-crl>
        QGp1bmlwZXIuY29tMB4XDTE0MDIyNzE0MTM1MloXDTE1MDIyNzE0MTM1MlowMDET
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        RDEuRiZNRNLeJpgN9YWkXLAZX2rASwy041EMmZ6KAkWUd3ZmXucfoLpdRemfuPii
        KQTpIM/rNrbrkuTmalezFoFS7mrxLXJAsfP1guVcD7sLCyjvegL8pRCCrU9xyKLF
        8u/Qz4s0x0uzcGYh0sd3iWj21+AtigSLdMD76/j/VzftQL8B1yp3vc1EZiowOwq4
        AwEAAaOCAW0wggFpMBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHppoyXF
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        NTOufhQsD2t4TYpEkzLEiZqSswdBOaPxPcJLQNW8Bw2xN+A9GX=
      </issuer-crl>
    </ownership-voucher>
    <owner-certificate>
      <certificate>
        MIIExTCCA62gAwIBAgIBATANBgkqhkiG9w0BAQsFADCBqjELMAkGA1UEBhMCVVMx
        EzARBgNVBAgTCkNhbGlmb3JuaWExEjAQBgNVBAcTCVN1bm55dmFsZTEZMBcGA1UE
        ChQQSnVuaXBlcl9OZXR3b3JrczEdMBsGA1UECxQUQ2VydGlmaWNhdGVfSXNzdWFu
        Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
        QGp1bmlwZXIuY29tMB4XDTE0MDIyNzE0MTM1MloXDTE1MDIyNzE0MTM1MlowMDET
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        RDEuRiZNRNLeJpgN9YWkXLAZX2rASwy041EMmZ6KAkWUd3ZmXucfoLpdRemfuPii
        ap1DgmS3IaYl/s4OOF8yzcYJprm8O7NyZp+Y9H1U/7Qfp97/KbqwCgkHSzOlnt0X
        KQTpIM/rNrbrkuTmalezFoFS7mrxLXJAsfP1guVcD7sLCyjvegL8pRCCrU9xyKLF
        8u/Qz4s0x0uzcGYh0sd3iWj21+AtigSLdMD76/j/VzftQL8B1yp3vc1EZiowOwq4
        KmORbiKU2GTGZkaCgCjmrWpvrYWLoXv/sf2nPLyK6YjiWsslOJtRO+KzRbs2B18C
        AwEAAaOCAW0wggFpMBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHppoyXF
        yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
        GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
        X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
        MjAOBgNVHQ8BAf8EBAMCAgQwQgYDVR0fBDswOTA3oDWgM4YxaHR0cDovL2NybC5q
        dW5pcGVyLm5ldD9jYT1KdW5pcGVyX1RydXN0X0FuY2hvcl9DQTANBgkqhkiG9w0B
```

```
        AQsFAAOCAQEAOuD7EBilqQcT3t2C4AXta1gGNNwdldLLw0jtk4BMiA9l//DZfskB
        2AaJtiseLTXsMF6MQwDs1YKkiXKLu7gBZDlJ6NiDwy1UnXhi2BDG+MYXQrc6p76K
        z3bsVwZlaJQCdF5sbggc1MyrsOu9QirnRZkIv3R8ndJH5K792ztLquulAcMfnK1Y
        NTOufhQsD2t4TYpEkzLEiZqSswdBOaPxPcJLQNW8Bw2xN+A9GX7WJzEbT/G7MUfo
        Sb+U2PVsQTDWEzUjVnG7vNWYxirnAOZ0OXEWWYxHUJntx6DsbXYuX7D1PkkNr7ir
        96DpOPtX7h8pxxGSDPBXIyvg02aFMphstQ==
      </certificate>
      <issuer-crl>
        Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
        MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
        ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
        yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
        WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
        MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
        GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
        X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
        MjAO==
      </issuer-crl>
    </owner-certificate>
  </device>
</devices>
```

6.2.5. Progress Notifications

The following example illustrates a device using the API to post a notification to the server. The device may send more than one notification to the server (e.g., to provide status updates). The YANG module defines only one notification type, bootstrap-complete. Other notification types may be defined through YANG augmentation.

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.

In this example, the device sends a notification indicating that it has completed bootstrapping off the data provided by the server. This example also illustrates the device sending its SSH host keys to the bootstrap server, which it might, for example, forward onto a downstream NMS component, so that it can subsequently authenticate the device when establishing a NETCONF over SSH connection to it.

A device providing its SSH host key or TLS server certificate is not needed when the device has an IDevID certificate [Std-802.1AR-2009] and is able to present the IDevID certificate as its SSH host key or TLS server certificate, when establishing a NETCONF or RESTCONF connection.

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

<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>AAAAB3NzaC1yc2EAAAADAQABAAABAQDeJMV8zrtsi8CgEsRCjCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStK
O2MwjE1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcCWAw1lOr9IDGDAuww6G45gLcHalHMmBtQx
fL3ZS5G76Fy6sA5vg7SLqQFPjXXft2CAhin8xwYRZy6r/
2N9PMJ2Dnepvq4H2DKqBIe340jWqEIuA7LvEJYql4unq4Iog+/
+CiumTkmQIWRgIoj4FCzYkO9NvRE6fOSLLf6gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</key-
data>
    </ssh-host-key>
    <ssh-host-key>
      <format>ssh-dsa</format>
      <key-
data>AAAAB3NzaC1yc2EAAAADAQABAAABAQDeJMV8zrtsi8CgEsRCjCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStK
O2MwjE1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcCWAw1lOr9IDGDAuww6G45gLcHalHMmBtQx
fL3ZS5G76Fy6sA5vg7SLqQFPjXXft2CAhin8xwYRZy6r/
2N9PMJ2Dnepvq4H2DKqBIe340jWqEIuA7LvEJYql4unq4Iog+/
+CiumTkmQIWRgIoj4FCzYkO9NvRE6fOSLLf6gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</key-
data>
    </ssh-host-key>
  </ssh-host-keys>
</input>


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

6.3. Artifact Examples

This section presents some examples for how the same information provided by the API can be packaged into stand alone artifacts. The encoding for these artifacts is the same as if an HTTP GET request had been sent to the RESTCONF URL for the specific resource.

Encoding these artifacts for use outside of the RESTCONF protocol extends their utility for other deployment scenarios, such as when a local DHCP server or a removable storage device is used. By way of example, this may be done to address an inability for the device to access an Internet facing bootstrap/redirect server, or just for a preference to use locally deployed infrastructure.

6.3.1. Signed Redirect Information

The following example illustrates how a redirect can be encoded into an artifact for use outside of the RESTCONF protocol. The redirect information is signed so that it is secure even when no transport-level security is provided.

```
<redirect-information xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-
bootstrap-server">
  <address>phs.example.com</address>
  <trust-anchor>
    WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc3OEgrRkYyRTFwdSt4ZVRJbVFFM
    lLQllsdWpOcjFTMnRLR05EMUc2OVJpK2FWNGw2NTdZNCtadVJMZgpRYjk
    zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot
    NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPCk1CMEdBMVVkRGd
    VEJiZ0JTWEdlbUEKMnhpRHVOTVkvVHFLNWd4cFJBZ1ZOYUU0cERZd05ER
    V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNSll6UG8zREF
    NQmdOVkhSTUJBZjhFCkFqQUFNQTRHQTFVZER3RUIvd1FFQXdJSGdEQnBC
    Z05WSFI4RVlqQmdNRjZnSXFBZ2hoNW9kSFJ3T2k4dlpYaGgKYlhCc1pTN
    WpiMjB2WlhoaGJYQnNaUzVqY215aU9LUTJNRFF4Q3pBSkJnTlZCQVlUQW
    QmdOVkJBWVRBbFZUTVJBd0RnWURWUVFLRXdkbAplR0Z0Y0d4bE1RNHdEQ
    MkF6a3hqUDlVQWtHR0dvS1U1eUc1SVR0Wm0vK3B0R2FieXVDMjBRd2kvZ
    25PZnpZNEhONApXY0pTaUpZK2xtYWs3RTRORUZXZS9RdGp4NUlXZmdvN2
    RJSUJQFRStS0Cg==
  </trust-anchor>
  <signature>
    SomeSignatureString
  </signature>
</redirect-information>
```

6.3.2. Signed Bootstrap Information

The following example illustrates how bootstrapping data can be encoded into an artifact for use outside of the RESTCONF protocol. The bootstrapping information is signed so that it is secure when no transport-level security is provided.

```
<bootstrap-information xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-
bootstrap-server">
  <boot-image>
    <name>
      boot-image-v3.2R1.6.img
    </name>
    <md5>
      SomeMD5String
    </md5>
    <sha1>
      SomeSha1String
    </sha1>
    <path>
      /some/path/to/raw/file
    </path>
    <signature>
      SomeSignatureString
    </signature>
```

```
  </boot-image>
  <configuration>
```

```
    <config>
      <!-- from ietf-system.yang -->
      <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
        <authentication>
          <user>
            <name>admin</name>
            <ssh-key>
              <name>admin's rsa ssh host-key</name>
              <algorithm>ssh-rsa</algorithm>
              <key-data>AAAAB3NzaC1yc2EAAAADAQABAAABAQDeJMV8zrtsi8CgEsRC
              jCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/O2Mwj
              E1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HMLlTRnoCVcC
              WAw1lOr9IDGDAuww6G45gLcHalHMmBtQxKnZdzU9kx/fL3ZS5G76Fy6sA5
              vg7SLqQFPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2DKqBIe340jWq
              EIuA7LvEJYql4unq4Iog+/+CiumTkmQIWRgIoj4FCzYkO9NvRE6fOSLLf6
              gakWVOZZgQ8929uWjCWlGlqn2mPibp2Go1</key-data>
            </ssh-key>
          </user>
        </authentication>
      </system>
      <!-- from ietf-netconf-server.yang -->
      <netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
        <call-home>
          <application>
            <name>config-mgr</name>
            <ssh>
              <endpoints>
                <endpoint>
                  <name>east-data-center</name>
                  <address>11.22.33.44</address>
                </endpoint>
                <endpoint>
                  <name>west-data-center</name>
                  <address>55.66.77.88</address>
                </endpoint>
              </endpoints>
              <host-keys>
                <host-key>my-call-home-x509-key</host-key>
              </host-keys>
            </ssh>
          </application>
        </call-home>
      </netconf-server>
    </config>
    <signature>
      SomeSignatureString
    </signature>
  </configuration>
```

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

6.3.3. Owner Certificate

The following example illustrates how the owner certificate, along with its CRL, can be encoded into an artifact for use outside of the RESTCONF protocol. As the Owner Certificate and CRL are already signed by the manufacturer, an additional owner signature is unnecessary.

```
<owner-certificate xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-
server">
  <certificate>
    MIIExTCCA62gAwIBAgIBATANBgkqhkiG9w0BAQsFADCBqjELMAkGA1UEBhMCVVMx
    EzARBgNVBAgTCkNhbGlmb3JuaWExEjAQBgNVBAcTCVN1bm55dmFsZTEZMBcGA1UE
    ChQQSnVuaXBlcl9OZXR3b3JrczEdMBsGA1UECxQUQ2VydGlmaWNhdGVfSXNzdWFu
    Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
    QGp1bmlwZXIuY29tMB4XDTE0MDIyNzE0MTM1MloXDTE1MDIyNzE0MTM1MlowMDET
    MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
    ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
    RDEuRiZNRNLeJpgN9YWkXLAZX2rASwy041EMmZ6KAkWUd3ZmXucfoLpdRemfuPii
    ap1DgmS3IaYl/s4OOF8yzcYJprm8O7NyZp+Y9H1U/7Qfp97/KbqwCgkHSzOlnt0X
    KQTpIM/rNrbrkuTmalezFoFS7mrxLXJAsfP1guVcD7sLCyjvegL8pRCCrU9xyKLF
    8u/Qz4s0x0uzcGYh0sd3iWj21+AtigSLdMD76/j/VzftQL8B1yp3vc1EZiowOwq4
    KmORbiKU2GTGZkaCgCjmrWpvrYWLoXv/sf2nPLyK6YjiWsslOJtRO+KzRbs2B18C
    AwEAAaOCAW0wggFpMBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHppoyXF
    yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
    WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
    MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
    GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
    X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
    MjAOBgNVHQ8BAf8EBAMCAgQwQgYDVR0fBDswOTA3oDWgM4YxaHR0cDovL2NybC5q
    dW5pcGVyLm5ldD9jYT1KdW5pcGVyX1RydXN0X0FuY2hvcl9DQTANBgkqhkiG9w0B
    AQsFAAOCAQEAOuD7EBilqQcT3t2C4AXta1gGNNwdldLLw0jtk4BMiA9l//DZfskB
    2AaJtiseLTXsMF6MQwDs1YKkiXKLu7gBZDlJ6NiDwy1UnXhi2BDG+MYXQrc6p76K
    z3bsVwZlaJQCdF5sbggc1MyrsOu9QirnRZkIv3R8ndJH5K792ztLquulAcMfnK1Y
    NTOufhQsD2t4TYpEkzLEiZqSswdBOaPxPcJLQNW8Bw2xN+A9GX7WJzEbT/G7MUfo
    Sb+U2PVsQTDWEzUjVnG7vNWYxirnAOZ0OXEWWYxHUJntx6DsbXYuX7D1PkkNr7ir
    96DpOPtX7h8pxxGSDPBXIyvg02aFMphstQ==
  </certificate>
  <issuer-crl>
    Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
    MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
    ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
    yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
    WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
    MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
    GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
    X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
    MjAO==
  </issuer-crl>
</owner-certificate>
```

6.3.4. Ownership Voucher

The following example illustrates how the ownership voucher, along with its CRL, can be encoded into an artifact for use outside of the RESTCONF protocol. As the Ownership Voucher and CRL are already

signed by the manufacturer, an additional owner signature is unnecessary.

```
<ownership-voucher xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-
server">
  <voucher>
    ChQQSnVuaXBlcl9OZXR3b3JrczEdMBsGA1UECxQUQ2VydGlmaWNhdGVfSXNzdWFu
    Y2UxGTAXBgNVBAMUEFRQTV9UcnVzdF9BbmNob3IxHTAbBgkqhkiG9w0BCQEWDmNh
    MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
    ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
    yh/JaftWYf7m3KBzOdg2MIHfBgNVHSMEgdcwgdSAFDSljCNmTN5b+CDujJLlyDal
    WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
    MBAGA1UEBxMJU3Vubnl2YWxlMRkwFwYDVQQKFBBKdW5pcGVyX05ldHdvcmtzMR0w
    GwYDVQQLFBRDZXJ0aWZpY2F0ZV9Jc3N1YW5jZTEZMBcGA1UEAxQQVFBNX1RydXN0
    X0FuY2hvcjEdMBsGCSqGSIb3DQEJARYOY2FAanVuaXBlci5jb22CCQDUbsEdTn5v
    MjAO
  </voucher>
  <issuer-crl>
    QGp1bmlwZXIuY29tMB4XDTE0MDIyNzE0MTM1MloXDTE1MDIyNzE0MTM1MlowMDET
    MBEGA1UEChQKVFBNX1ZlbmRvcjEZMBcGA1UEAxQQSnVuaXBlcl9YWFhYWF9DQTCC
    ASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANL5Mk5qFsVuqo+JmXWLmFxI
    RDEuRiZNRNLeJpgN9YWkXLAZX2rASwy041EMmZ6KAkWUd3ZmXucfoLpdRemfuPii
    KQTpIM/rNrbrkuTmalezFoFS7mrxLXJAsfP1guVcD7sLCyjvegL8pRCCrU9xyKLF
    8u/Qz4s0x0uzcGYh0sd3iWj21+AtigSLdMD76/j/VzftQL8B1yp3vc1EZiowOwq4
    AwEAAaOCAW0wggFpMBIGA1UdEwEB/wQIMAYBAf8CAQAwHQYDVR0OBBYEFHppoyXF
    WFPaoYGwpIGtMIGqMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTES
    NTOufhQsD2t4TYpEkzLEiZqSswdBOaPxPcJLQNW8Bw2xN+A9GX=
  </issuer-crl>
</ownership-voucher>
```

6.4. YANG Module

The bootstrap server's device-facing interface is normatively defined by the following YANG module:

```
<CODE BEGINS> file "ietf-zerotouch-bootstrap-server@2015-10-19.yang"

module ietf-zerotouch-bootstrap-server {

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

  import ietf-inet-types {          // RFC 6991
    prefix inet;
  }

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

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


  description
   "This module defines the southbound interface for Zero Touch
    bootstrap servers.

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

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

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

  revision "2015-10-19" {
    description
     "Initial version";
    reference
     "RFC XXXX: Zero Touch Provisioning for NETCONF Call Home";
  }


  grouping redirect-information-grouping {
    description
      "This container contains information the device may use
       to redirect it to another bootstrap server.";

    leaf address {
      type inet:host;
      mandatory true;
      description
       "The IP address or hostname of the bootstrap server
        the device should redirect to.";
    }
```

```
    leaf trust-anchor {
      type binary;
      mandatory true;
      description
        "A certificate that a device can use as a trust anchor to
         authenticate the bootstrap server it is being redirected
         to.  The binary certificate structure as specified by RFC
         5246, Section 7.4.6, i.e.,: opaque ASN.1Cert<1..2^24>;
        ";
      reference
        "RFC 5246: The Transport Layer Security (TLS)
                   Protocol Version 1.2";
    }

    leaf signature {
      type string;
      must "../../ownership-voucher";
      description
        "The signature over the concatenation of the previous leafs
         using the organization's private key.  Specifically,
         sign(name+md5+sha1+path), where simple string concatenation
         to join values is used, resulting in a single null-terminated
         string.";
      }


  }


  grouping bootstrap-information-grouping {

    container boot-image {
      description
        "It is intended that the device will fetch this container
         as a whole, as it contains values that need to be
         processed together.";
      leaf name {
        type string;
        mandatory true;
        description
          "The name of the image of software the device is expected
           to be running.";
      }
      leaf md5 {
        type string;
        mandatory true;
        description
          "The output of the MD5 hash algorithm over the image file.";
```

```
      }
      leaf sha1 {
        type string;
        mandatory true;
        description
          "The output of the SHA-1 hash algorithm over the image file.";
      }
      leaf path {
        type string;
        mandatory true;
        description
          "An absolute path to the boot-image file hosted on this
           Bootstrap server.";
      }
      leaf signature {
        type string;
        must "../../../ownership-voucher";
        description
          "The signature over the concatenation of the previous leafs
           using the organization's private key.  Specifically,
           sign(name+md5+sha1+path), where simple string concatenation
           to join values is used, resulting in a single null-terminated
           string.";
      }
    }

    container configuration {
      description
        "It is intended that the device will fetch this container
         as a whole, as its contents need to be processed together.";
      anyxml config {
        mandatory true;
        description
          "Any configuration data model known to the device.  It may
           contain Vendor-specific and/or standards-based data models.
           An example configuration using a couple IETF-defined data
           models is presented the Appendix of RFC XXXX.";
      }
      leaf signature {
        type string;
        must "../../../ownership-voucher";
        description
          "The signature over the concatenation of the previous leaf
           using the organization's private key.  Specifically,
           sign(config), where 'config' is treated as a single null-
           terminated string.";
      }
    }
```

```
  }


  grouping owner-certificate-grouping {

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

  grouping ownership-voucher-grouping {
    leaf voucher {
      type binary;
      mandatory true;
      description
        "A Vendor-specific encoding binding unique device
         identifiers to an owner identifier value matching the
         value encoded in the owner-certificate below.  An
         example format for a voucher is presented in the
         Appendix of RFC XXXX.";
    }
    leaf issuer-crl {
      type string;
      description
        "An optional CRL for the issuer used by the
         Vendor to sign Ownership Vouchers.  The CRL should be
         as up to date as possible.  This leaf is optional as
         it is primarily to support deployments where the device
         is unable to download the CRL from the CRL distribution
         point URLs listed in the Vendor's trust anchor
```

```
         certificate.";
    }
  }

  container devices {
    config false;
    description
      "A read-only list of device entries";
    list device {

      key unique-id;
      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.";
      }

      choice type {

        container redirect-information {
          uses redirect-information-grouping;
        }

        container bootstrap-information {
          uses bootstrap-information-grouping;
        }

      }

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

        when "../redirect-information/signature or ../bootstrap-information/*/
signature";
        //must "../owner-certificate and ../redirect-information/signature
or ../bootstrap-information/*/signature";
        must "../owner-certificate";

        uses ownership-voucher-grouping;
      }

      container owner-certificate {
        description
          "It is intended that the device will fetch this container
```

as a whole, as it contains values that need to be

```
           processed together.";

        when "../ownership-voucher";
        //must "../ownership-voucher and ../redirect-information/signature
or ../bootstrap-information/*/signature";

        uses owner-certificate-grouping;
      }


      action notification {
        input {
          leaf type {
            type enumeration {
              enum bootstrap-complete {
                description
                  "Indicates that the device successfully processed the
                   bootstrap data, that is currently running the specified
                   boot image and has committed the configuration.  At
                   this point, the device is ready to be managed by an
                   external NMS system.  The device is never expected
                   access the bootstrap server again, unless reset to
                   its factory default again.";
              }
            }
            mandatory true;
          }
          leaf message {
            type string;
            description
              "A human-readable value.";
          }
          container ssh-host-keys {
            list ssh-host-key {
              when "../type = bootstrap-complete";
              leaf format {
                type enumeration {
                  enum ssh-dss;
                  enum ssh-rsa;
                }
                mandatory true;
              }
              leaf key-data {
                type string;
                mandatory true;
              }
            }
          }
```

```
        }
```

```
      } // end action
    }
  }

}
```

<CODE ENDS>

7. Security Considerations

7.1. Immutable storage for trust anchors

Devices MUST ensure that all their trust anchor certificates, including those for the Owner Certificate and Ownership Voucher, 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.

7.2. Real time clock

The solution for signed data includes validating Owner Certificates and Ownership Vouchers, each of which may contain expirations. Further, the solution includes using a CRLs, which also require freshness. Device implementations should take care to ensure the devices have a reliable clock when processing signed data.

7.3. Entropy loss over time

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

7.4. 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.

8. IANA Considerations

Editor Note: this section needs to be rewritten to use the redirect and bootstrap information types (see Section 2.4).

8.1. Zero Touch Information DHCP Options

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

8.1.1. DHCP v4 Option

Tag: XXX

Name: Zero Touch Information

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

Code	Len			
XXX	n	svr1	svr2	...

Reference: RFC XXXX

8.1.2. DHCP v6 Option

Tag: YYY

Name: Zero Touch Information

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

Code	Len			
YYY	n	svr1	svr2	...

Reference: RFC XXXX

9. Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): David Harrington, Dean Bogdanovic, Martin Bjorklund, Max Pritikin, Stephen Hanna, Wes Hardaker, Russ Mundy, Reinaldo Penno, Randy Presuhn, Juergen Schoenwaelder.

Special thanks goes to Steve Hanna, Russ Mundy, and Wes Hardaker for brainstorming the original I-D's solution during the IETF 87 meeting in Berlin.

10. References

10.1. Normative References

[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>.

[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>.

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

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

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

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

10.2. Informative References

[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>.

[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>.

Appendix A. Examples

A.1. Ownership Voucher

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

A tree diagram describing an Ownership Voucher:

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

The YANG module for this example voucher:

<CODE BEGINS> file "ietf-zerotouch-ownership-voucher@2015-10-19.yang"

```
module ietf-zerotouch-ownership-voucher {

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-zerotouch-ownership-voucher";
  prefix "ztov";

  import ietf-yang-types { prefix yang; }

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

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


  description
   "This module defines the format for a ZeroTouch ownership voucher,
    which is produced by Vendors, relayed by Bootstrap Servers, and
    consumed by devices.  The purpose of the voucher is to enable a
```

```
    device to ascertain the identity of its rightful owner, as
    certified by its Vendor.

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

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

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

  revision "2015-10-19" {
    description
     "Initial version";
    reference
     "RFC XXXX: Zero Touch Provisioning for NETCONF Call Home";
  }

  // top-level container
  container voucher {
    description
      "A voucher, containing the owner's identifier, a list of
       device's unique identifiers, information on when the
       voucher was created, when it might expire, and the
       vendor's signature over the above values.";
    leaf owner-id {
      type string;
      mandatory true;
      description
        "A Vendor-assigned value for the rightful owner of the
         devices enumerated by this voucher.  The owner-id value
         must match the value in the owner-certificate below";
    }
    leaf-list unique-id {
      type string;
      min-elements 1;
      description
        "The unique identifier (e.g., serial-number) for a device.
         The value must match the value in the device's IDevID
         certificate.  A device uses this value to determine if
         the voucher applies to it.";
    }
    leaf created-on {
```

```
        type yang:date-and-time;
        mandatory true;
        description
          "The date this voucher was created";
      }
      leaf expires-on {
        type yang:date-and-time;
        description
          "The date this voucher expires, if at all.  Use of this
           value requires that the device has access to a trusted
           real time clock";
      }
      leaf signature {
        type string;
        mandatory true;
        description
          "The signature over the concatenation of all the previous
           values";
      }
    }
  }
```

<CODE ENDS>

Appendix B. Change Log

B.1. ID to 00

- Major structural update; the essence is the same. Most every section was rewritten to some degree.
- Added a Use Cases section
- Added diagrams for "Actors and Roles" and "NMS Precondition" sections, and greatly improved the "Device Boot Sequence" diagram
- Removed support for physical presence or any ability for Configlets to not be signed.
- Defined the Zero Touch Information DHCP option
- Added an ability for devices to also download images from Configuration Servers
- Added an ability for Configlets to be encrypted

- Now Configuration Servers only have to support HTTP/S - no other schemes possible

B.2. 00 to 01

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

B.3. 01 to 02

- Replaced the need for a Configuration Signer with the ability for each NMS to be able to sign its own configurations, using 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

B.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.

B.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".

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Joe Clarke
Cisco Systems

EMail: jclarke@cisco.com

Mikael Abrahamsson
T-Systems

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