

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

K. Watsen
Juniper Networks
J. Clarke
Cisco Systems
M. Abrahamsson
T-Systems
July 6, 2015

Zero Touch Provisioning for NETCONF Call Home (ZeroTouch)
draft-ietf-netconf-zerotouch-03

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

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

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

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Editor's Note	3
1.2.	Use Cases	4
1.3.	Terminology	4
1.4.	Tree Diagrams	5
2.	High-level Design	5
2.1.	Design Overview	5
2.2.	Interactions	8
3.	Bootstrap Server	11
3.1.	Northbound Interface	11
3.2.	Southbound Interface	11
4.	Device	20
4.1.	Factory Default State	20
4.2.	Boot Sequence	21
5.	Network Management System (NMS)	25
5.1.	Overview	25
5.2.	Precondition	25
5.3.	Connection Handling	26
6.	Security Considerations	26
6.1.	Entropy loss over time	26
6.2.	Serial Numbers	26
7.	IANA Considerations	27
7.1.	ZeroTouch Information DHCP Options	27
7.1.1.	DHCP v4 Option	27
7.1.2.	DHCP v6 Option	27
8.	Acknowledgements	27
9.	References	28
9.1.	Normative References	28
9.2.	Informative References	28
Appendix A.	Examples	29
A.1.	Ownership Voucher	29
A.2.	Bootstrap Server's API	31
A.3.	Bootstrap Configuration	31
Appendix B.	Change Log	33
B.1.	ID to 00	33
B.2.	00 to 01	33
B.3.	01 to 02	33
B.4.	02 to 03	34

1. Introduction

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

The solution presented herein enables a device to securely obtain a bootstrapping configuration from the network without any operator input. Significantly, this configuration may configure the device to securely call home using NETCONF Call Home [[draft-ietf-netconf-call-home](#)].

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

1.1. Editor's Note

This draft defines a solution that differs from the solution presented in [[draft-pritikin-anima-bootstrapping-keyinfra](#)]. Yet it is this author's opinion that it should be possible for both solutions to co-exist simultaneously. How to integrate them is discussed in this section.

Already the ANIMA draft defines in Section #3 the progression of first searching link-local, then the local network (e.g., DHCP options or DNS service records), and finally trying Internet-based resources. This progression makes good sense and, in general, there may be other searching options before, after, or in between the ones listed here (e.g., USB flash drive). The important aspect to this list of options is that they're ordered so as to give priority to the "closest" option.

It seems right that a device try all comparably-secure bootstrapping options at each stage of the searching progression. For instance, an integrated solution might try all comparably-secure bootstrapping options when at the link-local stage, and again try all comparably-secure bootstrapping options when at the local network stage, and yet again try all the comparably-secure bootstrapping options when at the Internet stage.

By trying all comparably-secure options at each stage, it 1) minimizes the number of times a device needs to reconfigure its networking, 2) prevents a "further away" solution for one option

snuffing out a closer solution for a closer solution for another option, and 3) avoids having devices reach out to remote options unnecessarily, and thereby reduce tracking.

Bootstrapping options that are not comparably-secure should not be integrated as described above. Specifically, all secure bootstrapping options should be attempted prior to attempting any unsecure option. For instance, a device should prefer a secure option with a remote server over an unsecure option available on the link-local network.

The solution presented below does not reflect the thoughts mentioned in this note. Specifically, it has no provision for using a link-local address or DNS service discovery. Instead, it only assumes the use of a DHCP server, potentially with DHCP options to direct the device to use resources on the local network.

1.2. 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 services it can use to bootstrap off of.

- o Connecting to a locally administered network

This use-case covers all other scenarios (including link-local) 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 discovered, 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 case.

1.3. Terminology

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

1.4. Tree Diagrams

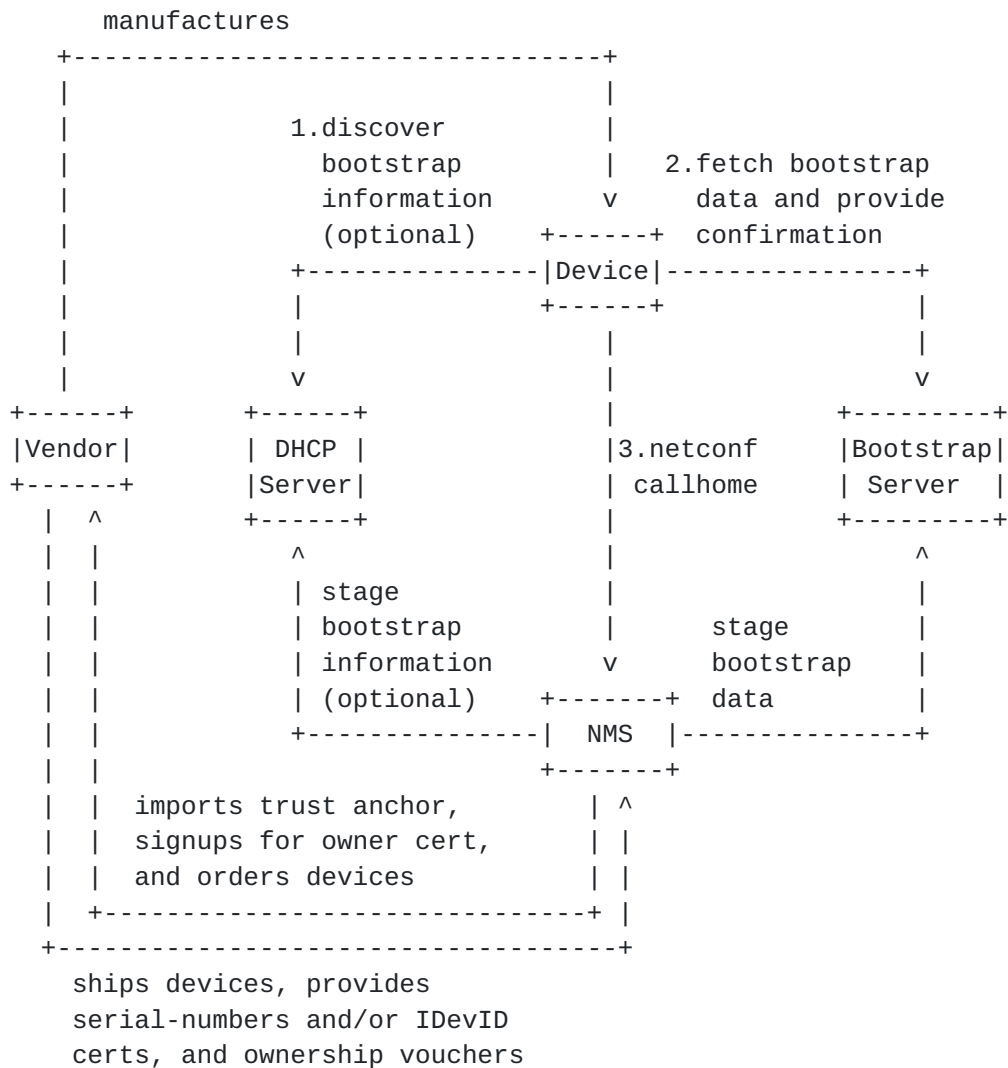
A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

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

2. High-level Design

2.1. Design Overview

The following diagram illustrates the overall solution presented in this draft. Note that some of the interactions illustrated below occur at different times, only the numbered interactions (1-3) occur at the time a device is bootstrapping itself.



The boxes in this diagram are described next. A sequence diagram explaining the various calls follows in [Section 2.2](#).

o Vendor

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

o Device

The devices supporting NETCONF ZeroTouch will only attempt the bootstrapping process when booting with its factory default

configuration. As illustrated above, the bootstrapping process consists of three interactions:

1. When joining the network, the device will attempt to configure IP networking from a DHCP server. If the device is able to reach a DHCP server, it may discover additional bootstrapping information. The additional bootstrapping information consists of one or more additional Bootstrap Servers the device should try to connect to.
2. The device sequentially processes its list of Bootstrap Servers, prioritizing any that might have been learned from the DHCP server. Once the device has successfully configured itself using the bootstrapping information, it notifies the bootstrapping server for monitoring purposes.
3. Assuming the bootstrapping information configures the device appropriately, the device will initiate a NETCONF Call Home connection [[draft-ietf-netconf-call-home](#)].

More information about Devices is in [Section 4](#).

o DHCP Server

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

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

o Bootstrap Server

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

Bootstrap Servers may be deployed on the public Internet or on a local network. Devices may be preconfigured with a list of well-known Bootstrap Servers. Additional Bootstrap Servers (i.e. not in the device's preconfigured list) must be discovered from a DHCP server.

How Bootstrap Servers are deployed is out of scope of this draft, but there are a couple points worth noting. Firstly, it is expected that Internet based Bootstrap Servers will initially be hosted by Vendors, whilst waiting for 3rd-party servers to become available. Secondly, it is expected that locally administrated networks with in-house solutions might bundle the Bootstrap Server into another system (e.g., the NMS), where having the features integrated can streamline various workflows.

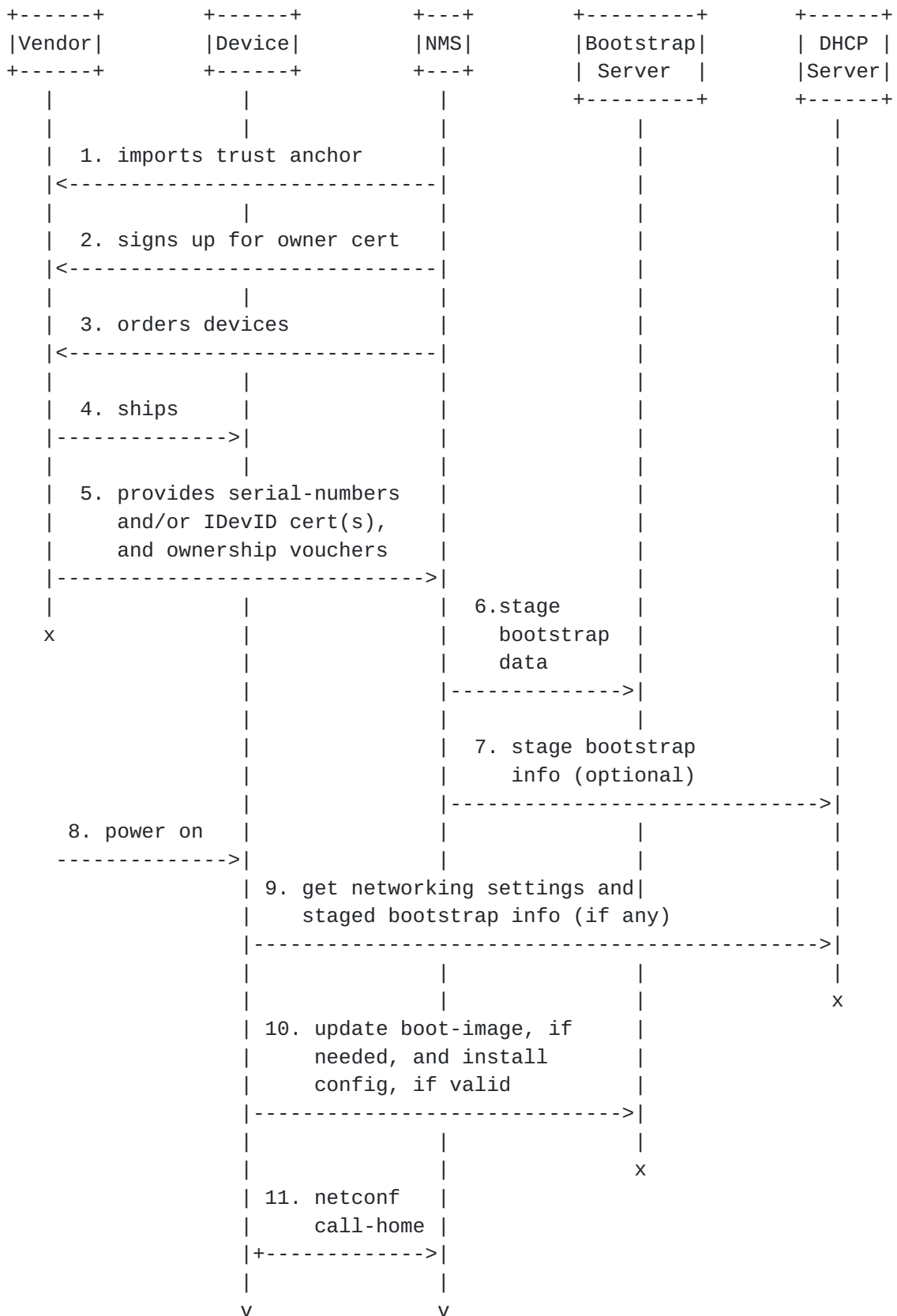
More information about Bootstrap Servers is in [Section 3](#).

- o Network Management System

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

[2.2](#). Interactions

The following diagram illustrates the interactions between the entities described in the previous section. Note that the interactions can be roughly categorized as those that occur before a device powers on and those that occur after a device powers on.



These interactions are described below.

1. An organization, upon deciding to deploy a Vendor's devices for NETCONF ZeroTouch, would import into its NMS the IDevID trust anchor certificate from the Vendor. This certificate is later used by the NMS to authenticate device identities during NETCONF call home connections.
2. An organization needs to sign up to a Vendor-provided ZeroTouch program. This program entails the Vendor providing a signed Owner certificate to the organization (depicted here), as well as a commitment to sign Ownership Vouchers for future device orders (interaction #5).
3. Subsequently, the organization may place orders to the Vendor for devices supporting ZeroTouch. The ordering process may entail an explicit request for ZeroTouch support, as the Vendor providing the files in step #5 may not be enabled by default.
4. The Vendor ships the devices to the various addresses specified in the device order. For example, to an organization's inventory warehouse, where the devices are stored in batches to supply internal requests. In another example, the devices may be shipped to their final deployment destinations.
5. In order to support ZeroTouch, the Vendor sends to the organization information about the devices it shipped. This information may be sent to the organization via email or a portal site. The information includes the serial-number and/or IDevID certificate, for each device, as well as one more Ownership Vouchers, assigning ownership for the devices to the organization.
6. In anticipation for the devices performing the ZeroTouch process, the NMS configures the Bootstrap Server. This This configuration includes everything a device needs to securely connect to the NMS.
7. For deployments where the DHCP server can be customized, the NMS may configure the DHCP server to provide the device a list of additional Bootstrap Servers to consider, in addition to those the device knows of by default. This customization can be configured at a global level in the DHCP server, as it is not dependent on the type of device in any way.
8. At some point, the device powers on and, when having its factory default configuration, initiates the ZeroTouch process.

9. The device obtains from the DHCP server a dynamic network assignment. The device may also at this time discover a list of additional bootstrap servers, as optionally configured by the NMS in step #7.
10. The device iterates over its list of Bootstrap Servers, until it can successfully bootstrap its initial configuration. If it is unable to bootstrap an initial configuration, the device boots as normal. If the staged information directs the device to load a new image, it does so and reboots. If the device reboots, it continues to have a factory default configuration state, which then bring it back to this state, when it would then have the correct image. The device then loads the staged configuration into its running datastore, after validating that the configuration was signed by its rightful owner, as designated by the Ownership Voucher.
11. Assuming the bootstrapping information configures the device appropriately, the device will initiate a NETCONF Call Home [[draft-ietf-netconf-call-home](#)] connection to the NMS, which then takes over the on-going management of the device.

3. Bootstrap Server

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

3.1. Northbound Interface

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

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

3.2. Southbound Interface

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

A tree diagram describing the Bootstrap Server's southbound interface follows:

```

module: ietf-zerotouch-bootstrap-server
  +--ro devices
    +--ro device* [unique-id]
      +--ro unique-id          string
      +--ro ownership-voucher
      | +--ro voucher          binary
      | +--ro issuer-crl?     string
      +--ro owner-certificate
      | +--ro certificate      string
      | +--ro issuer-crl?     string
      +--ro boot-image!
      | +--ro name             string
      | +--ro md5              string
      | +--ro sha1             string
      | +--ro path             string
      | +--ro signature        string
      +--ro configuration
        +--ro config
        +--ro signature        string

  rpcs:
    +---x notification
      +---w input
        +---w unique-id          string
        +---w type                enumeration
        +---w message?           string
        +---w ssh-host-keys!
          +---w ssh-host-key*
            +---w format          enumeration
            +---w key-data        string

```

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

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

<CODE BEGINS> file "ietf-zerotouch-bootstrap-server@2015-07-06.yang"

```

module ietf-zerotouch-bootstrap-server {

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

```


organization

"IETF NETCONF (Network Configuration) Working Group";

contact

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

description

"This module defines the southbound interface for ZeroTouch Bootstrap Servers.

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

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

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

```
revision "2015-07-06" {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: Zero Touch Provisioning for NETCONF Call Home";  
}
```

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



```
}

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

container owner-certificate {
  // presense?
  description
    "It is intended that the device will fetch this container
    as a whole, as it contains values that need to be
    processed together.";
  leaf certificate {
    type string;
    mandatory true;
    description
      "This is an X.509 certificate, signed by a Vendor, for
      a business organization. This certificate must encode a
      Vendor-assigned value identifying the organization. This
      identifier must match the owner identifier encoded in
      the Ownership Voucher.";
  }
  leaf issuer-crl {
```



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

container boot-image {
    presence
        "Only present when boot image information has been configured";
    description
        "It is intended that the device will fetch this container
        as a whole, as it contains values that need to be
        processed together.";
    leaf name {
        type string;
        mandatory true;
        description
            "The name of the image of software the device is expected
            to be running.";
    }
    leaf 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;
        mandatory true;
    }
}
```



```
    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;
    mandatory true;
    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.";
  }
}
/*
action notification {
  input {
    leaf type {
      type enumeration {
        enum boot-image-missing {
          description
            "Indicates that the device got an error when trying to
            access the provided URL";
        }
        enum boot-image-invalid {
          description
            "Indicates that the device had a problem processing the
            boot-image file (corruption)";
        }
        enum image-name-mismatch {
          description
```



```
        "Indicates that the processed boot-image contains a name
        other than provided";
    }
    enum voucher-invalid {
        description
        "Indicates that the device had a problem processing the
        voucher (chain verification failed, revoked crl)";
    }
    enum owner-cert-invalid {
        description
        "Indicates that the device had a problem processing the
        voucher (chain verification failed, revoked crl)";
    }
    enum owner-id-mismatch {
        description
        "Indicates that the owner-id in the voucher does not
        match the one inside the owner-cert";
    }
    enum signature-invalid {
        description
        "Indicates that the signature could not be verified
        using the owner-cert";
    }
    enum bootstrap-complete {
        description
        "Indicates that the device successfully processed the
        bootstrap data. At this point, the device is running
        the required boot-image and configuration. A device
        is expected to only send this notification once,
        assuming it does not receive an error in the HTTP
        response from the Bootstrap Server.";
    }
}
mandatory true;
}
leaf message {
    type string;
    description
    "A human-readable value.";
}
container ssh-host-keys {
    presence "Only present for bootstrap-complete messages.";
    list ssh-host-key {
        leaf format {
            type enumeration {
                enum ssh-dss;
                enum ssh-rsa;
            }
        }
    }
}
```



```
        mandatory true;
    }
    leaf key-data {
        type string;
        mandatory true;
    }
}
}
}
} // end action
*/
}
}

rpc notification {
    input {
        leaf unique-id {
            type string;
            mandatory true;
        }
        leaf type {
            type enumeration {
                enum boot-image-missing {
                    description
                        "Indicates that the device got an error when trying to
                        access the provided URL";
                }
                enum boot-image-invalid {
                    description
                        "Indicates that the device had a problem processing the
                        boot-image file (corruption)";
                }
                enum image-name-mismatch {
                    description
                        "Indicates that the processed boot-image contains a name
                        other than provided";
                }
                enum voucher-invalid {
                    description
                        "Indicates that the device had a problem processing the
                        voucher (chain verification failed, revoked crl)";
                }
                enum owner-cert-invalid {
                    description
                        "Indicates that the device had a problem processing the
                        voucher (chain verification failed, revoked crl)";
                }
                enum owner-id-mismatch {
```

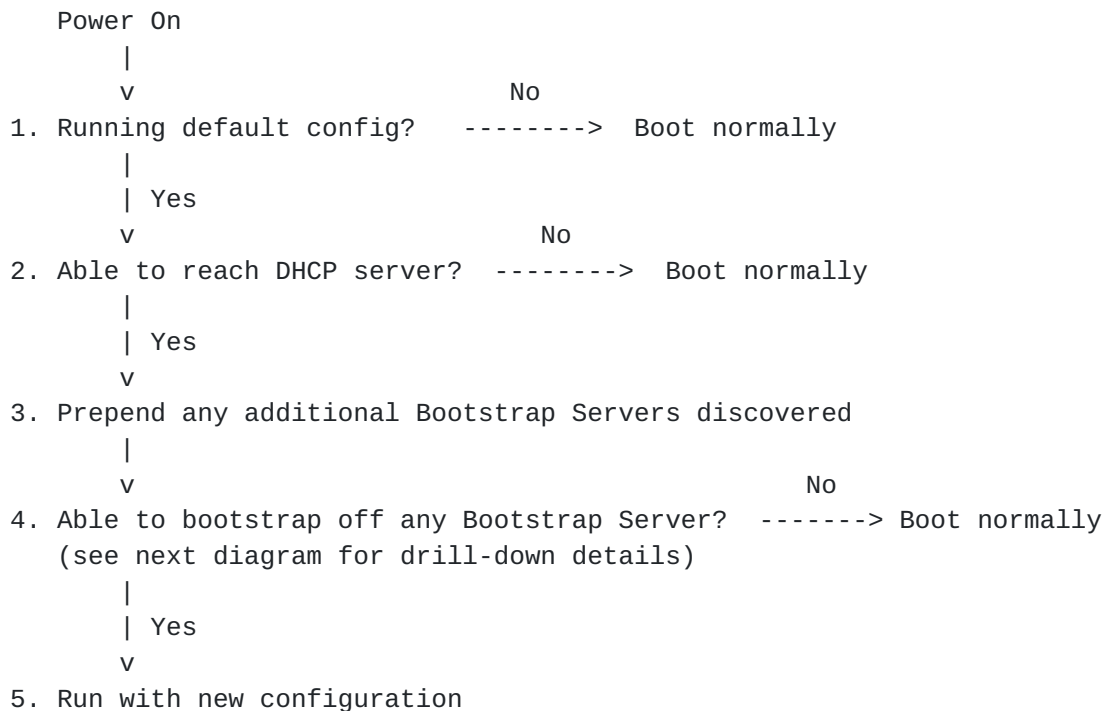


```
        description
        "Indicates that the owner-id in the voucher does not
        match the one inside the owner-cert";
    }
    enum signature-invalid {
        description
        "Indicates that the signature could not be verified
        using the owner-cert";
    }
    enum bootstrap-complete {
        description
        "Indicates that the device successfully processed the
        bootstrap data. At this point, the device is running
        the required boot-image and configuration. A device
        is expected to only send this notification once,
        assuming it does not receive an error in the HTTP
        response from the Bootstrap Server.";
    }
    }
    mandatory true;
}
leaf message {
    type string;
    description
    "A human-readable value that might provide useful information";
}
container ssh-host-keys {
    presence "Only present for bootstrap-complete messages.";
    list ssh-host-key {
        leaf format {
            type enumeration {
                enum ssh-dss;
                enum ssh-rsa;
            }
            mandatory true;
        }
        leaf key-data {
            type string;
            mandatory true;
        }
    }
}
}
} // end rpc notification
}

<CODE ENDS>
```


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

[4.2.](#) Boot Sequence



These interactions are described next.

1. When the device powers on, it first checks to see if it is running the factory default configuration. If it is running a modified configuration, then it boots normally.
2. The device tries to obtain a dynamic network assignment from a DHCP server. If it is unable to reach a DHCP server, it boots normally.
3. If the DHCP server's offer includes the ZeroTouch Information DHCP option defined in [Section 7.1](#), the device prepends the specified Bootstrap Servers to its factory default list.
4. The device iterates over its list of Bootstrap Servers, as described in the next section. If it is unable to bootstrap itself off any of the servers, it boots normally.
5. If the device was able to bootstrap itself off any of the Bootstrap Servers, it runs with the new configuration merged into its running datastore.

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


```

    Connect to port 443
    |
    v
1. Able to validate server certificate? -----> Exit
    |
    | Yes
    v
2. Able to validate ownership voucher? -----> Post notification and exit
    |
    | Yes
    v
3. Able to validate owner certificate? -----> Post notification and exit
    |
    | Yes
    v
4. Able to validate boot image info? -----> Post notification and exit
    |
    | Yes
    v
5. Need to install boot image? -----> Install and reboot
    |
    | Yes
    v
6. Able to validate configuration? -----> Post notification and exit
    |
    | Yes
    v
7. Merge configuration into running datastore
    |
    v
8. Post bootstrap complete notification and exit

```

These interactions are described next.

1. As part of the HTTPS connection, the device will need to authenticate the server certificate presented by the Bootstrap Server. The device authenticates the server certificate using path-validation to one of its preconfigured Bootstrap Server trust anchors. If the device is unable to authenticate the server's certificate, it abandons this Bootstrap Server and exits.
2. The device downloads the ownership voucher from the Bootstrap Server. The device validates the voucher is signed by its Vendor, using its preconfigured trust anchor for device ownership vouchers. The device also validates that its unique identifier is listed by the voucher. If the device is unable to validate the voucher or can not find its unique identifier listed, it

posts a notification message to that effect and abandons this Bootstrap Server.

3. The device downloads the owner certificate from the Bootstrap Server. The device validates that this certificate is signed by its Vendor, using path-validation to its preconfigured trust anchor for owner certificates. The device also validates that the organization identifier is the same as listed in the ownership voucher, validated in step #2. If the device is unable to validate the certificate or the owner identifier does not match, it posts a notification message to that effect and abandons this Bootstrap Server.
4. The device tries to download the boot image information. If no boot image information is available, it skips the remainder of this step. Otherwise, the device validates the boot image information using the public key from the owner certificate obtained in step #3. If it is unable to authenticate the boot image information, it posts a notification message to that effect and abandons this Bootstrap Server.
5. The device checks if the specified boot-image name matches what the device is currently running. If there is a mismatch, device downloads the new image from the Bootstrap Server and installs it. It is expected that the device will reboot itself in order to activate the new image and, further, that doing so preserves its factory default state such that it will return to this same check again, but then running the correct image. If the device is unable to install the boot-image, it posts a notification message to that effect and abandons this Bootstrap Server.
6. The device downloads the configuration from the Bootstrap Server and validates the configuration using the public key from the owner certificate obtained in step #3. If it is unable to authenticate the configuration, it posts a notification message to that effect and abandons this Bootstrap Server.
7. The device merges the configuration into its running datastore. It is expected that this configuration will provide the information necessary for the device to establish a secure NETCONF connection to its NMS using NETCONF Call Home ([\[draft-ietf-netconf-call-home\]](#)).
8. The device posts a bootstrap completion notification message to the Bootstrap Server and exits.

2. In order for the NMS to validate that a specific device connecting to it is legitimate, it MUST have a list of expected unique device identifiers (e.g., serial-numbers). The unique identifier encoded into the device's IDevID certificate MUST match one of the expected identifiers in order for a device to be considered legitimate.
3. The NMS must have login credentials for each device. These credentials may be, for instance, a private key used for SSH or TLS client authentication. It is expected that a device is able to authenticate the NMS's credentials by virtue of the configuration it downloads from the Bootstrap Server. These private-keys SHOULD be stored securely, such that they can not be easily compromised.

5.3. Connection Handling

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

6. Security Considerations

6.1. Entropy loss over time

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

6.2. Serial Numbers

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

7. IANA Considerations

7.1. ZeroTouch Information DHCP Options

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

7.1.1. DHCP v4 Option

Tag: XXX

Name: Zero Touch Information

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

Code	Len
XXX	n svr1 svr2 ...

Reference: RFC XXXX

7.1.2. DHCP v6 Option

Tag: YYY

Name: Zero Touch Information

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

Code	Len
YYY	n svr1 svr2 ...

Reference: RFC XXXX

8. Acknowledgements

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

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

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), October 2010.
- [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", [RFC 6187](#), March 2011.
- [Std-802.1AR-2009]
IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [[draft-ietf-netconf-call-home](#)]
Watsen, K., "NETCONF Call Home (work in progress)", October 2014, <<https://tools.ietf.org/html/draft-ietf-netconf-call-home-04>>.
- [[draft-ietf-netconf-restconf](#)]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [draft-ietf-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>>.

9.2. Informative References

- [[draft-pritikin-anima-bootstrapping-keyinfra](#)]
Pritikin, M., Behringer, M., and S. Bjarnason, "Bootstrapping Key Infrastructures", [draft-pritikin-anima-bootstrapping-keyinfra-01](#) (work in progress), 2015, <<https://tools.ietf.org/html/draft-pritikin-anima-bootstrapping-keyinfra>>.

[Appendix A](#). Examples

[A.1](#). Ownership Voucher

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

A tree digram describing an Ownership Voucher:

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

The YANG module for this example voucher:

<CODE BEGINS> file "ietf-zerotouch-ownership-voucher@2015-07-06.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 "2015-07-06" {
  description
    "Initial version";
  reference
    "RFC XXXX: Zero Touch Provisioning for NETCONF Call Home";
}

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



```
    type yang:date-and-time;
    description
        "The date this voucher expires, if any";
}
leaf signature {
    type string;
    mandatory true;
    description
        "The signature over the concatenation of all the previous
        values";
}
}
}
```

<CODE ENDS>

[A.2.](#) Bootstrap Server's API

['\ ' line wrapping added for formatting only]

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

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

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

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

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

[A.3.](#) Bootstrap Configuration

This example illustrates a configuration enabling secure NETCONF call-home using standards-based YANG modules.


```
<?xml version="1.0"?>
<configuration>

  <!-- from ietf-system.yang -->
  <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
    <authentication>
      <user>
        <name>admin</name>
        <ssh-key>
          <name>admin's rsa ssh host-key</name>
          <algorithm>ssh-rsa</algorithm>
          <key-data>AAAAB3NzaC1yc2EAAAADAQABAAQDeJMV8zrtsi8CgEsRC
            jCzfve2m6zD3awSBPrh7ICggLQvHVbPL89eHLuecStKL3HrEgXaI/02Mwj
            E1lG9YxLzeS5p2ngzK61vikUSqfMukeBohFTrDZ8bUtrF+HML1TRnoCVcC
            WAw1l0r9IDGDAuww6G45gLcHalHMmBtQxKnZdzU9kx/fL3ZS5G76Fy6sA5
            vg7SLqQFPjXXft2CAhin8xwYRZy6r/2N9PMJ2Dnepvq4H2DKqBIe340jWq
            EIuA7LvEJYq14unq4Iog+/+CiumTkmQIWRgIoJ4FCzYk09NvRE6f0SLLf6
            gakWVOZZgQ8929uWjCWlG1qn2mPibp2Go1</key-data>
          </ssh-key>
        </user>
      </authentication>
    </system>

    <!-- from ietf-netconf-server.yang -->
    <netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
      <call-home>
        <application>
          <name>config-mgr</name>
          <ssh>
            <endpoints>
              <endpoint>
                <name>east-data-center</name>
                <address>11.22.33.44</address>
              </endpoint>
              <endpoint>
                <name>west-data-center</name>
                <address>55.66.77.88</address>
              </endpoint>
            </endpoints>
            <host-keys>
              <host-key>my-call-home-x509-key</host-key>
            </host-keys>
          </ssh>
        </application>
      </call-home>
    </netconf-server>
  </configuration>
```


[Appendix B](#). Change Log

[B.1](#). ID to 00

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

[B.2](#). 00 to 01

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

[B.3](#). 01 to 02

- o Replaced the need for a Configuration Signer with the ability for each NMS to be able to sign its own configurations, using Vendor signed Ownership Vouchers and Owner certificates.

- o Renamed Configuration Server to Bootstrap Server, a more representative name given the information devices download from it.
- o Replaced the concept of a Configlet by defining a southbound interface for the Bootstrap Server using YANG.
- o Removed the IANA request for the boot-image and configuration media types

B.4. 02 to 03

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

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](mailto:mikael.abrahamsson@t-systems.se)

