     Zero Touch Provisioning for NETCONF or RESTCONF based Management
                  draft-ietf-netconf-zerotouch-17

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

   o "OPTION_V4_ZEROTOUCH_REDIRECT" --> the option code assigned for
      the "DHCPv4 Zero Touch Option" option

   o "OPTION_V6_ZEROTOUCH_REDIRECT" --> the option code assigned for
      the "DHCPv6 Zero Touch Option" option

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

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

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

   o "2017-09-11" --> the publication date of this draft

Please update the following references to reflect their final RFC
assignments:

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

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

o [Appendix A]. Change Log

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 [https://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 March 15, 2018.

Copyright Notice

Table of Contents

1. Introduction

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

   This document defines a bootstrapping strategy enabling devices to
   securely obtain bootstrapping data with no installer action beyond
   physical placement and connecting network and power cables. The
   ultimate goal of this document is to enable a secure NETCONF

[RFC6241] or RESTCONF [RFC8040] connection to a deployment specific
network management system (NMS).

This document primarily regards physical devices, where the setting
of the device's initial state, described in Section 5.1, occurs
during the device's manufacturing process. However, the zerotouch
solution may be extensible to virtual machines whereby, for instance,
the host-system has an IDevID certificate and some ability to verify
the virtual machine's image before loading it and also provision a
just-in-time IDevID certificate. Details for how this may be
accomplished can be defined in future work.

1.1. Use Cases

o Device connecting to a remotely administered network

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

o Device connecting to a locally administered network

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

Conceptual workflows for how zerotouch might be deployed are provided
in Appendix A.

1.2. Terminology

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

Artifact: The term "artifact" is used throughout to represent any of
      the three artifacts defined in Section 3 (Zero Touch Information,
      Ownership Voucher, and Owner Certificate). These artifacts
      collectively provide all the bootstrapping data a device may use.

Bootstrapping Data: The term "bootstrapping data" is used throughout
      this document to refer to the collection of data that a device

   may obtain during the bootstrapping process. Specifically, it
   refers to the three artifacts defined in Section 3.

Bootstrap Server: The term "bootstrap server" is used within this
   document to mean any RESTCONF server implementing the YANG module
   defined in Section 7.4.

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

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

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

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

Onboarding Information: The term "onboarding information" is used
   herein to refer to one of the two types of 'zero touch
   information' (see term) defined in this document, the other being
   'redirect information'. Specifically, onboarding information is
   defined by the 'onboarding-information' YANG-data structure in
   Section 6.3.

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

Owner Certificate: The term "owner certificate" is used in this
   document to represent an X.509 certificate that binds an owner
   identity to a public key, which a device can use to validate a
   signature over the zero touch information artifacts. The owner
   certificate is one of the three bootstrapping artifacts described
   in Section 3.

Ownership Voucher: The term "ownership voucher" is used in this
   document to represent the voucher artifact defined in

[I-D.ietf-anima-voucher]. The ownership voucher is used to
assign a device to an owner. The ownership voucher is one of the
three bootstrapping artifacts described in Section 3.

Redirect Information: The term "redirect information" is used herein
to refer to one of the two types of 'zero touch information' (see
term) defined in this document, the other being 'onboarding
information'. Specifically, redirect information is defined by
the 'redirect-information' YANG-data structure in Section 6.3.

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

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

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

Zero Touch Information: The term "zero touch information" is used
generally herein to refer either redirect information or
onboarding information. Zero touch information is one of the
three bootstrapping artifacts described in Section 3.

1.3. Requirements Language

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

1.4. Tree Diagram Notation

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

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

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

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

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. Types of Bootstrapping Information

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

2.1. Redirect Information

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

Redirect information is YANG modeled data formally defined by the
"redirect-information" container in the YANG module presented in
Section 6.3. This container has the tree diagram shown below.
Please see Section 1.4 for tree diagram notation.

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

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

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

establish a secure connection to a bootstrap server, the data is
implicitly trusted, and does not need to be signed.

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

How devices process redirect information is described more formally
in Section 5.5.

2.2. Onboarding Information

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

Bootstrap information is YANG modeled data formally defined by the
"onboarding-information" container in the YANG module presented in
Section 6.3. This container has the tree diagram shown below.
Please see Section 1.4 for tree diagram notation.

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

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

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

How devices process onboarding information is described more formally
in Section 5.6.

3. Artifacts

This document defines three artifacts that can be made available to
devices while they are bootstrapping. Each source of bootstrapping
information specifies a means for providing each of the artifacts
defined in this section (see Section 4).

3.1. Zero Touch Information

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

The zero touch information artifact is a PKCS#7 SignedData structure,
as specified by Section 9.1 of [RFC2315], encoded using ASN.1
distinguished encoding rules (DER), as specified in ITU-T X.690. The
PKCS#7 structure MUST contain JSON-encoded content conforming to the
YANG module specified in Section 6.3.

In order for the zero touch information artifact to be trusted when
conveyed over an untrusted transport, the PKCS#7 structure MUST also
contain a 'signerInfo' structure, as described in Section 9.1 of
[RFC2315], containing a signature generated over the content using
the private key associated with the owner certificate (Section 3.2).
In order to simplify the verification process, the PKCS#7 structure
MUST also contain the signing X.509 certificate.

3.2. Owner Certificate

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

The owner certificate is used by a device to verify the signature for
the zero touch information artifact (Section 3.1), the device SHOULD

also have received, as described in Section 3.4. In particular, the
device verifies signature using the public key in the owner
certificate over the content contained within the zero touch
information artifact.

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

The owner certificate PKCS#7 structure MUST contain the owner
certificate itself, as well as all intermediate certificates leading
up to the 'pinned-domain-cert' certificate specified in the ownership
voucher. The owner certificate artifact MAY optionally include the
trust anchor certificate.

Additionally, in order to support devices deployed on private
networks, the owner certificate PKCS#7 structure MAY also contain
suitably fresh CRLs [RFC5280] and/or OCSP Responses [RFC6960].
Having these revocation objects stapled to the owner certificate
precludes the need for the device to have to download them
dynamically using the CRL distribution point or an OCSP responder
specified in the associated certificates.

3.3. Ownership Voucher

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

More specifically, the ownership voucher is used to verify the owner
certificate (Section 3.2) that the device SHOULD have also received,
as described in Section 3.4. In particular, the device verifies that
the owner certificate has a chain of trust leading to the trusted
certificate included in the ownership voucher, even if it is itself
(e.g., self-signed certificate).

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

3.4. Artifact Groupings

This section lists all the possible bootstrapping artifacts, but only
certain groupings of these artifacts make sense to return in the
various bootstrapping situations described in this document. These
groupings are:

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

Signed Information, without revocations: The grouping is useful
when signed information is needed, because it's obtained from
an untrusted source, and it cannot be processed provisionally,
and yet either revocations are not needed or they can be
obtained dynamically.

Signed Information, with revocations: The grouping is useful when
signed information is needed, because it's obtained from an
untrusted source, and it cannot be processed provisionally, and
revocations are needed and cannot be obtained dynamically.

The artifacts associated with these groupings are described below:

Grouping	Zero Touch Information	Ownership Voucher	Owner Certificate
Unsigned Information	Yes, no sig	No	No
Signed Information, without revocations	Yes, with sig	Yes, without revocations	Yes, without revocations
Signed Information, with revocations	Yes, with sig	Yes, with revocations	Yes, with revocations

4. Sources of Bootstrapping Data

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

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

4.1. Removable Storage

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

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

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

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

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

Artifact to File Mapping:

   Zero Touch Information: Mapped to a file containing the binary
      artifact described in Section 3.1 (e.g., zerotouch-
      information.pk7).

   Owner Certificate: Mapped to a file containing the binary
      artifact described in Section 3.2 (e.g., owner-
      certificate.pk7).

   Ownership Voucher: Mapped to a file containing the binary
      artifact described in Section 3.3 (e.g., ownership-
      voucher.pk7).

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

4.2. DNS Server

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

Using a DNS server may be a compelling option for deployments having
existing DNS infrastructure, as it enables a touchless bootstrapping

   option that does not entail utilizing an Internet based resource
   hosted by a 3rd-party.

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

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

   From an artifact perspective, since a DNS server presents resource
   records (Section 3.2.1 of [RFC1035]), the bootstrapping artifacts
   need to be presented as resource records. The three artifacts
   defined in Section 3 are mapped to resource records below.

   Artifact to Resource Record Mapping:

      Zero Touch Information: Mapped to a TXT record called "zt-info"
         containing the base64-encoding of the binary artifact described
         in Section 3.1.

      Owner Certificate: Mapped to a TXT record called "zt-cert"
         containing the base64-encoding of the binary artifact described
         in Section 3.2.

      Ownership Voucher: Mapped to a TXT record called "zt-voucher"
         containing the base64-encoding of the binary artifact described
         in Section 3.3.

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

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

4.3. DHCP Server

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

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

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

   However, unlike other sources of bootstrapping data described in this
   document, the DHCP protocol (especially DHCP for IPv4) is limited in
   the amount of data that can be conveyed, to the extent that signed
   data cannot be communicated. This means only unsigned redirect
   information can be conveyed. Since the redirect information is
   unsigned, it SHOULD NOT include the optional trust anchor
   certificate, as the device would have to discard it anyway.

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

      Zero Touch Information: This artifact is not supported directly.
         Instead, the essence of redirect information (not onboarding
         information) is mapped to the DHCP fields described in
         Section 8.

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

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

4.4. Bootstrap Server

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

   Unlike any other source of bootstrap data described in this document,
   a bootstrap server is not only a source of data, but it can also
   receive data from devices using the YANG-defined 'update-progress'
   action defined in the YANG module (Section 7.4). The data sent from

devices both enables visibility into the bootstrapping process (e.g.,
warnings and errors) as well as provides potentially useful
completion status information (e.g., the device's SSH host-keys).

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

Using a bootstrap server as a source of bootstrapping data is a
compelling option as it MAY use transport-level security, in lieu of
signed data, which may be easier to deploy in some situations.
Additionally, the bootstrap server is able to receive progress
updates from devices, which may be critical to some deployments
(e.g., the passing of the device's SSH host keys).

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

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

From an artifact perspective, since a bootstrap server presents data
as a YANG-modeled data, the bootstrapping artifacts need to be mapped
to nodes in the YANG module. The three artifacts defined in
Section 3 are mapped to bootstrap server nodes defined in Section 7.4
below.

Artifact to Bootstrap Server Node Mapping:

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

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

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

While RESTCONF servers typically support a nested hierarchy of
resources, zero touch bootstrap servers only need to support the
paths /device and /device/update-progress. The device processing
instructions provided in Section 5.3 only uses these two URLs.

5. Device Details

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

5.1. Initial State

```
+-------------------------------------------------------------+
|                         <device>                            |
|                                                             |
| +---------------------------------------------------------+ |
| |                   <read-only storage>                   | |
| |                                                         | |
| | 1. IDevID cert & associated intermediate certificate(s) | |
| | 2. list of trusted Internet based bootstrap servers     | |
| | 3. list of trust anchor certs for bootstrap servers     | |
| | 4. trust anchor cert for verifying ownership vouchers   | |
| +---------------------------------------------------------+ |
|                                                             |
|                 +---------------------+                     |
|                 |   <secure storage>  |                     |
|                 |                     |                     |
|                 |  5. private key     |                     |
|                 +---------------------+                     |
|                                                             |
+-------------------------------------------------------------+
```
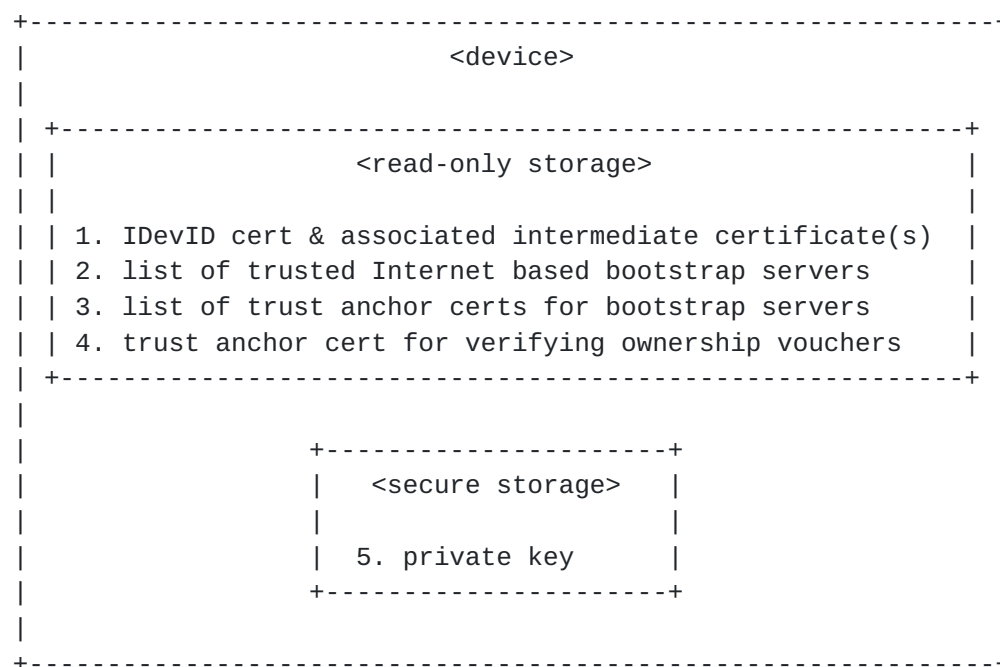
Each numbered item below corresponds to a numbered item in the
diagram above.

1.   Devices MUST possess an initial device identifier (IDevID), as
     defined in [Std-802.1AR-2009]. The IDevID is an X.509
     certificate, encoding e.g., the device's serial number and
     hardware manufacturer. The device MUST also possess any
     intermediate certificates between the IDevID certificate and the
     IDevID trust anchor certificate, which is provided to prospective
     owners separately (e.g., Appendix A.1).

2.   Devices that support loading bootstrapping data from an Internet-
     based bootstrap server (see Section 4.4) MUST possess:

   * A configured list of trusted bootstrap servers. Consistent
      with redirect information (Section 2.1, each bootstrap server
      MAY be identified by its hostname or IP address, and an
      optional port.

   * A configured list of trust anchor certificates that can be
      used for X.509 certificate path validation ([RFC6125],
      Section 6) on the bootstrap server's TLS server certificate.

   3. Devices that support loading signed data (see Section 1.2) MUST
       possess the manufacturer's trust anchor certificate for
       validating ownership vouchers.

   4. Devices MUST possess 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 the bootstrapping strategy defined in
   this document MUST support the boot sequence described in this
   section.

```
    Power On
        |
        v                                 No
 1. Zerotouch bootstrapping configured  -------->  Boot normally
        |
        | Yes
        v
 2. For each supported source of bootstrapping data,
    try to load bootstrapping data from the source
        |
        |
        v                                 Yes
 3. Able to bootstrap from any source?  ----> Run with new configuration
        |
        | No
        v
 4. Loop and/or wait for manual provisioning.
```
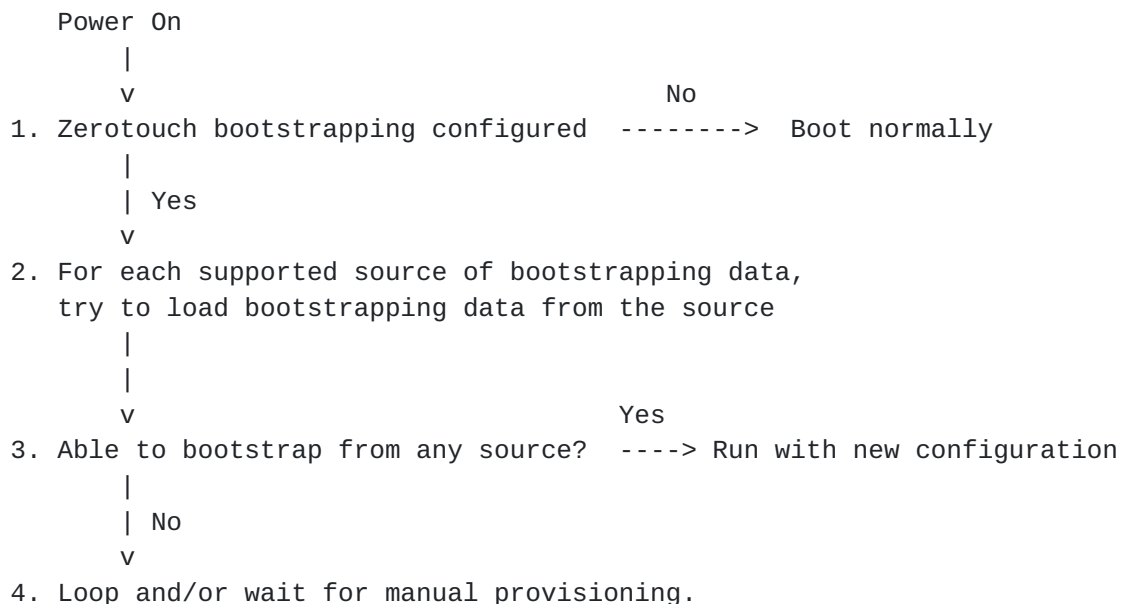
   Each numbered item below corresponds to a numbered item in the
   diagram above.

   1. When the device powers on, it first checks to see if zerotouch
       bootstrapping is configured, as is expected to be the case for

      the device's preconfigured state. If zerotouch bootstrapping is
      not configured, then the device boots normally.

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

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

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

5.3. Processing a Source of Bootstrapping Data

   This section describes a recursive algorithm that devices can use to,
   ultimately, obtain onboarding information. The algorithm is
   recursive because sources of bootstrapping data MAY return redirect
   information, which causes the algorithm to run again, for the newly
   discovered sources of bootstrapping information. An ANBF-like
   expression that captures all possible sequences of bootstrapping
   information is "(redirect information)* onboarding information".
   That is, zero or more redirect information responses, followed by one
   bootstrap information response.

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

```
                           Untrusted Source   Trusted Source
      Kind of Bootstrapping Data    Can Provide?     Can Provide?

      Unsigned Redirect Info    :       Yes+             Yes
      Signed Redirect Info      :       Yes              Yes*
      Unsigned Onboarding Info  :        No              Yes
      Signed Onboarding Info    :       Yes              Yes*
```

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

      The '*' above denotes that, while possible, it is generally
      unnecessary for a trusted source to return signed data. In
      fact, it's only needed when the '+' case occurs.

   The recursive algorithm uses a conceptual global-scoped variable
   called "trust-state". The trust-state variable is initialized to
   FALSE. The ultimate goal of this algorithm is for the device to

process onboarding information (Section 2.2) while the trust-state
variable is TRUE.

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

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

When accessing a bootstrap server, the device SHOULD only access its
top-level resource, to obtain all the data staged for it in a single
GET request.

For any source of bootstrapping data (e.g., Section 4), if the data
is signed and the device is able to validate the signed data using
the algorithm described in Section 5.4, then the device MUST set
trust-state to TRUE, else the device MUST set trust-state to FALSE.
Note, this is worded to cover the special case when signed data is
returned even from a trusted bootstrap server.

If the data is onboarding information (not redirect information), and
trust-state is FALSE, the device MUST exit the recursive algorithm
(as this is not allowed, per the figure above), returning to the
state machine described in Section 5.2. Otherwise, the device MUST
attempt to process the onboarding information as described in
Section 5.6. In either case, success or failure, the device MUST
exit the recursive algorithm, returning to the state machine
described in Section 5.2, the only difference being in how it
responds to the "Able to bootstrap from any source?" conditional
described in the figure in the section.

If the data is redirect information, the device MUST process the
redirect information as described in Section 5.5. This is the
recursion step, it will cause to device to reenter this algorithm,
but this time the data source will most definitely be a bootstrap
server, as that is all redirect information is able to redirect a
device to.

5.4. Validating Signed Data

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

   Whenever there is signed data, the device MUST also be provided an
   ownership voucher and an owner certificate. How all the needed
   artifacts are provided for each source of bootstrapping data is
   defined in Section 4.

   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), which may entail using additional
   intermediate certificates attached to the ownership voucher. If the
   device has an accurate clock, it MUST ensure that the ownership
   voucher was created in the past (i.e., 'created-on' < now). If the
   'expires-on' leaf is present, the device MUST verify that the
   ownership voucher has not yet expired (i.e., now < 'expires-on'),
   which requires an accurate clock. The device MUST verify that the
   ownership voucher's 'assertion' value is acceptable (e.g., some
   devices may only accept the assertion value 'verified'). The device
   MUST verify that the ownership voucher specifies the device's serial
   number in the 'serial-number' leaf. If the 'idevid-issuer' leaf is
   present, the device MUST verify that the value is set correctly. If
   the authentication of the ownership voucher is successful, the device
   extracts the 'pinned-domain-certificate' node, an X.509 certificate,
   that is needed to verify the owner certificate in the next step.

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

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

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

[5.5](). Processing Redirect Information

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

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

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

If the redirect information is trusted (e.g., trust-state is TRUE), and the bootstrap server entry contains a trust anchor certificate, then the device MUST authenticate the bootstrap server using X.509 certificate path validation ([[RFC6125], Section 6]()) to the specified trust anchor. If the device is unable to authenticate the bootstrap server to the specified trust anchor, the device MUST NOT attempt a provisional connection to the bootstrap server (i.e., by blindly accepting its server certificate).

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

[5.6](). Processing Onboarding Information

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

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

First the device MUST determine if the image it is running satisfies the specified boot image criteria (e.g., name and/or fingerprint match). If it does not, the device MUST download (using the supplied URI), verify, and install the specified boot image, and then reboot.

To verify the boot image, the device MUST check that the boot image
file matches the fingerprint (e.g., sha256) supplied by the
bootstrapping information. Upon rebooting, the device MUST still be
in its initial state, causing the bootstrapping process to run again,
which will eventually come to this very point, but this time the
device's running image will satisfy the specified criteria, and thus
the device will move to processing the next step.

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

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

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

At this point, the device has completely processed the bootstrapping
data and is ready to be managed. If the device obtained the
bootstrap information from a trusted bootstrap server, the device
MUST post the 'bootstrap-complete' progress update now, using the
bootstrap server's 'update-progress' action.

At this point the device is running its initial configuration.
Notably, if NETCONF Call Home or RESTCONF Call Home [RFC8071] is
configured, the device initiates trying to establish a call home
connection at this time.

6. The Zero Touch Information Artifact

This section defines a YANG 1.1 [RFC7950] module that is used to
define the data model for the zero touch information artifact
described in Section 3.1. Examples illustrating this artifact in use
are provided in Section 6.2.

6.1. Data Model Overview

The following tree diagram provides an overview of the data model for
the zero touch information artifact. The syntax used for this tree
diagram is described in Section 1.4.

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

6.2. Example Usage

This section presents examples for how the zero touch information
artifact (Section 3.1) can be encoded into a document that can be
distributed outside the bootstrap server's RESTCONF API.

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

```
   <redirect-information
     xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-information">
     <bootstrap-server>
       <address>phs1.example.com</address>
       <port>8443</port>
       <trust-anchor>base64encodedvalue==</trust-anchor>
     </bootstrap-server>
     <bootstrap-server>
       <address>phs2.example.com</address>
       <port>8443</port>
       <trust-anchor>base64encodedvalue==</trust-anchor>
     </bootstrap-server>
     <bootstrap-server>
       <address>phs3.example.com</address>
       <port>8443</port>
       <trust-anchor>base64encodedvalue==</trust-anchor>
     </bootstrap-server>
   </redirect-information>
```

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

```
   <onboarding-information
     xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-information">
     <boot-image>
       <name>boot-image-v3.2R1.6.img</name>
       <sha256>base64encodedvalue==</sha256>
       <uri>file:///some/path/to/raw/file </uri>
     </boot-image>
     <configuration-handling>merge</configuration-handling>
     <configuration>
       <!-- from ietf-system.yang -->
       <system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
         <authentication>
           <user>
             <name>admin</name>
             <authorized-key>
               <name>admin's rsa ssh host-key</name>
               <algorithm>ssh-rsa</algorithm>
               <key-data>base64encodedvalue==</key-data>
             </authorized-key>
           </user>
         </authentication>
       </system>
       <!-- from ietf-netconf-server.yang -->
       <netconf-server
         xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
```

```
            <call-home>
              <netconf-client>
                <name>config-mgr</name>
                <ssh>
                  <endpoints>
                    <endpoint>
                      <name>east-data-center</name>
                      <address>east.config-mgr.example.com</address>
                    </endpoint>
                    <endpoint>
                      <name>west-data-center</name>
                      <address>west.config-mgr.example.com</address>
                    </endpoint>
                  </endpoints>
                  <host-keys>
                    <host-key>
                      <name>certificate</name>
                      <certificate>builtin-idevid-cert</certificate>
                    </host-key>
                  </host-keys>
                  <client-cert-auth>
                    <trusted-ca-certs>
                      deployment-specific-ca-certs
                    </trusted-ca-certs>
                    <trusted-client-certs>
                      explicitly-trusted-client-certs
                    </trusted-client-certs>
                  </client-cert-auth>
                </ssh>
                <connection-type>
                  <periodic>
                    <idle-timeout>300</idle-timeout>
                    <reconnect-timeout>60</reconnect-timeout>
                  </periodic>
                </connection-type>
                <reconnect-strategy>
                  <start-with>last-connected</start-with>
                  <max-attempts>3</max-attempts>
                </reconnect-strategy>
              </netconf-client>
            </call-home>
          </netconf-server>
        </configuration>
      </onboarding-information>
```

6.3. YANG Module

   The zero touch information artifact is normatively defined by the
   YANG module defined in this section.

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

```
<CODE BEGINS> file "ietf-zerotouch-information@2017-09-11.yang"
module ietf-zerotouch-information {
  yang-version "1.1";

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

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

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

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

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

  description
   "This module defines the data model for the Zero Touch Information
    artifact defined by RFC XXXX: Zero Touch Provisioning for NETCONF
    or RESTCONF based Management.

    Copyright (c) 2017 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or without
    modification, is permitted pursuant to, and subject to the license
    terms contained in, the Simplified BSD License set forth in Section
    4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
```

```
     (http://trustee.ietf.org/license-info).

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

   revision "2017-09-11" {
     description
      "Initial version";
     reference
      "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
       Management";
   }

   rc:yang-data zerotouch-information {

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

       container redirect-information {
         description
           "Redirect information is described in Section 2.1 in
            RFC XXXX.  Its purpose is to redirect a device to
            another bootstrap server.";
         reference
           "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF
            based Management";

         list bootstrap-server {
           key address;
           description
             "A bootstrap server entry.";

           leaf address {
             type inet:host;
             mandatory true;
             description
              "The IP address or hostname of the bootstrap server the
               device should redirect to.";
           }
           leaf port {
             type inet:port-number;
             default 443;
```

```
             description
              "The port number the bootstrap server listens on.  If no
               port is specified, the IANA-assigned port for 'https'
               (443) is used.";
           }
           leaf trust-anchor {
             type binary;
             description
               "An X.509 v3 certificate structure as specified by RFC
                5280, Section 4, encoded using ASN.1 distinguished
                encoding rules (DER), as specified in ITU-T X.690.  A
                certificate that  device can use as the trust anchor
                to authenticate the bootstrap server the device is
                being redirected to.";
             reference
               "RFC 5280:
                  Internet X.509 Public Key Infrastructure Certificate
                  and Certificate Revocation List (CRL) Profile.
                ITU-T X.690:
                   Information technology - ASN.1 encoding rules:
                   Specification of Basic Encoding Rules (BER),
                   Canonical Encoding Rules (CER) and Distinguished
                   Encoding Rules (DER).";
           }
         }
       }

       container onboarding-information {
         description
           "Bootstrap information is described in Section 2.2 in
            RFC XXXX.  Its purpose is to provide the device
            everything it needs to bootstrap itself.";
         reference
           "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF
            based Management";

         container boot-image {
           description
             "Specifies criteria for the boot image the device MUST
              be running.";

           leaf name {
             type string;
             mandatory true;
             description
               "The name of a software image that the device MUST
                be running in order to process the remaining nodes.";
           }
```

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

          leaf configuration-handling {
            type enumeration {
              enum merge {
                 description
                   "Merge configuration into the running datastore.";
              }
              enum replace {
                 description
                   "Replace the existing running datastore with the
                    passed configuration.";
              }
            }
            description
              "This enumeration indicates how the server should process
               the provided configuration.  When not specified, the device
               MAY determine how to process the configuration using other
               means (e.g., vendor-specific metadata).";
          }

          leaf pre-configuration-script {
```

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

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

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

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

       No attempt is made to standardize the contents, running context,
       or programming language of the script.  The contents of the
       script are considered specific to the vendor, product line,
       and/or model of the device.

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

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

       If the exit status code is greater than zero, then the device
       should assume that the script had a soft error, which the
```

       script believes does not affect manageability. If the device
       obtained the bootstrap information from a bootstrap server,
       it SHOULD send a 'script-warning' notification message.

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

}
<CODE ENDS>

7. The Zero Touch Bootstrap Server API

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

7.1. API Overview

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

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

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

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

7.2. Query Parameters

   Bootstrap servers may optionally support the query parameters defined
   in this section.

   o The "os-name" Query Parameter

         The "os-name" query parameter indicates the name of the
         operating system the device is running. This parameter is only
         needed if it is possible for the device, as identified by its
         <unique-id> to run more than one type of operating system.
         This query parameter MUST be supported by servers needing to
         support such devices.

         HTTP Methods: GET, HEAD

         Capability URI: urn:ietf:params:restconf:capability:os-name:1.0

   o The "os-version" Query Parameter

         The "os-version" query parameter indicates the version of the
         operating system the device is running. This parameter enables
         the server to dynamically generate an optimal response for the
         device negating, for instance, the need for a potentially
         expensive boot-image update. This query parameter MUST be
         supported by servers wanting to support such use-cases.

         HTTP Methods: GET, HEAD

         Capability URI: urn:ietf:params:restconf:capability:os-
         version:1.0

   o The "circuit-id" Query Parameter

         The "circuit-id" query parameter, along with the "remote-id"
         query parameter, enables a device to propagate information
         collected (e.g., via DHCP Option 82) to the bootstrap server.
         This information enables the bootstrap server to return a
         customized response independent of the device's unique
         identifier.

HTTP Methods: GET, HEAD

Capability URI: urn:ietf:params:restconf:capability:circuit-
id:1.0

o The "remote-id" Query Parameter

The "remote-id" query parameter, along with the "circuit-id"
query parameter, enables a device to propogate information
collected (e.g., via DHCP Option 82) to the bootstrap server.
This information enables the bootstrap server to return a
customized response independent of the device's unique
identifier.

HTTP Methods: GET, HEAD

Capability URI: urn:ietf:params:restconf:capability:remote-
id:1.0

o The "nonce" Query Parameter

The "nonce" query parameter enables a device without an
accurate clock to obtain a one-time use voucher, as opposed to
a never-expiring voucher that it would otherwise receive. This
query parameter is only useful for cases where the device is
connected to an untrusted bootstrap server, when signed data
must be returned. The bootstrap server uses backend APIs not
described in this document to dynamically obtain the one-time
use voucher from the device's manufacturer.

HTTP Methods: GET, HEAD

Capability URI: urn:ietf:params:restconf:capability:nonce:1.0

7.3. Example Usage

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

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

REQUEST

['\' line wrapping added for formatting only]

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

RESPONSE

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

```
<device
  xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <unique-id>123456789</unique-id>
  <zerotouch-information>base64encodedvalue==</zerotouch-information>
  <owner-certificate>base64encodedvalue==</owner-certificate>
  <ownership-voucher>base64encodedvalue==</ownership-voucher>
</device>
```

The following example illustrates a device using the API to post a
progress update to a bootstrap server. Illustrated below is the
'bootstrap-complete' message, but the device may send other progress
updates to the server while bootstrapping (e.g., to provide status
updates). In this message, the device is sending both its SSH host
keys and TLS server certificate, which the bootstrap server may, for
example, pass to an NMS, as discussed in Appendix A.3.

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

Note that the bootstrap server MUST NOT process a progress update
from a device without first authenticating the device. This is in
contrast to when a device is fetching data from the server, a read-
only operation, in which case device authentication is not strictly
required (e.g., when sending signed information).

REQUEST

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

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

<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <device
    xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
      <unique-id>123456789</unique-id>
      <update-progress>
        <update-type>bootstrap-complete</update-type>
        <message>example message</message>
        <ssh-host-keys>
          <ssh-host-key>
            <format>ssh-rsa</format>
            <key-data>base64encodedvalue==</key-data>
          </ssh-host-key>
          <ssh-host-key>
            <format>ssh-dss</format>
            <key-data>base64encodedvalue==</key-data>
          </ssh-host-key>
        </ssh-host-keys>
        <trust-anchors>
          <trust-anchor>
            <protocol>netconf-ssh</protocol>
            <protocol>netconf-tls</protocol>
            <protocol>restconf-tls</protocol>
            <protocol>netconf-ch-ssh</protocol>
            <protocol>netconf-ch-tls</protocol>
            <protocol>restconf-ch-tls</protocol>
            <certificate>base64encodedvalue==</certificate>
          </trust-anchor>
        </trust-anchors>
      </update-progress>
    </device>
  </action>
</rpc>


RESPONSE
--------

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

Server: example-server

7.4. YANG Module

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

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

```
<CODE BEGINS> file "ietf-zerotouch-bootstrap-server@2017-09-11.yang"
module ietf-zerotouch-bootstrap-server {
  yang-version "1.1";

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

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

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

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

    Copyright (c) 2017 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

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

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

  revision "2017-09-11" {
    description
     "Initial version";
    reference
```

```
     "RFC XXXX: Zero Touch Provisioning for NETCONF or RESTCONF based
      Management";
  }

  // typedefs

  typedef pkcs7 {
    type binary;
    description
      "A PKCS #7 SignedData structure, as specified by Section 9.1
       in RFC 2315, encoded using ASN.1 distinguished encoding rules
       (DER), as specified in ITU-T X.690.";
     reference
       "RFC 2315:
          PKCS #7: Cryptographic Message Syntax Version 1.5.
        ITU-T X.690:
          Information technology - ASN.1 encoding rules:
          Specification of Basic Encoding Rules (BER),
          Canonical Encoding Rules (CER) and Distinguished
          Encoding Rules (DER).";
  }

  // protocol accessible nodes

  list device {
    key unique-id;
    config false;
    description
      "A device record containing bootstrapping data for the device
       with the specified identifier.  This top-level node provides
       a device with all the data it needs in a single request.";
    reference
      "RFC XXXX: Zero Touch Provisioning for NETCONF or
       RESTCONF based Management";

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

    leaf zerotouch-information {
      type pkcs7;
      mandatory true;
      description
        "A 'zerotouch-information' artifact, as described in Section
```

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

    leaf owner-certificate {
      type pkcs7;
      description
        "An unsigned PKCS #7 SignedData structure, as specified by
         Section 9.1 in RFC 2315, encoded using ASN.1 distinguished
         encoding rules (DER), as specified in ITU-T X.690.

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

         X.509 certificates and CRLs are described in RFC 5280.
         OCSP Responses are described in RFC 6960.";
      reference
       "RFC 2315:
           PKCS #7: Cryptographic Message Syntax Version 1.5.
        RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile.
        RFC 6960:
           X.509 Internet Public Key Infrastructure Online
           Certificate Status Protocol - OCSP.
        ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }

    leaf ownership-voucher {
      type pkcs7;
      must "../owner-certificate" {
        description
          "An owner certificate must be present whenever an ownership
```

```
            voucher is presented.";
        }
        description
          "A 'voucher' artifact, as described in Section 5 of
           I-D.ietf-anima-voucher.  The voucher informs the device
           who it's 'owner' is.  The voucher encodes the device's
           serial number, so that the device can be ensured that
           the voucher applies to it.  The voucher is signed by
           the device's manufacturer or delegate.";
        reference
          "I-D.etf-anima-voucher:
             Voucher and Voucher Revocation Profiles for Bootstrapping
             Protocols";
      }

      action update-progress {
        input {
          leaf update-type {
            type enumeration {
              enum bootstrap-initiated {
                description
                  "Indicates that the device has just accessed the
                   bootstrap server.  The 'message' field below MAY
                   contain any additional information that the
                   manufacturer thinks might be useful.";
              }
              enum parsing-warning {
                description
                  "Indicates that the device had a non-fatal error when
                   parsing the response from the bootstrap server.  The
                   'message' field below SHOULD indicate the specific
                   warning that occurred.";
              }
              enum parsing-error {
                description
                  "Indicates that the device encountered a fatal error
                   when parsing the response from the bootstrap server.
                   For instance, this could be due to malformed encoding,
                   the device expecting signed data when only unsigned
                   data is provided, because the ownership voucher didn't
                   include the device's unique identifier, or because the
                   signature didn't match.  The 'message' field below
                   SHOULD indicate the specific error.  This update type
                   also indicates that the device has abandoned trying to
                   bootstrap off this bootstrap server.";
              }
              enum boot-image-warning {
                description
```

```
              "Indicates that the device encountered a non-fatal
               error condition when trying to install a boot-image.
               A possible reason might include a need to reformat a
               partition causing loss of data.  The 'message' field
               below SHOULD indicate any warning messages that were
               generated.";
          }
          enum boot-image-error {
            description
              "Indicates that the device encountered an error when
               trying to install a boot-image, which could be for
               reasons such as a file server being unreachable,
               file not found, signature mismatch, etc.  The
               'message' field SHOULD indicate the specific error
               that occurred.  This update type also indicates
               that the device has abandoned trying to bootstrap
               off this bootstrap server.";
          }
          enum pre-script-warning {
            description
              "Indicates that the device obtained a greater than
               zero exit status code from the script when it was
               executed.  The 'message' field below SHOULD indicate
               both the resulting exit status code, as well as
               capture any stdout/stderr messages the script may
               have produced.";
          }
          enum pre-script-error {
            description
              "Indicates that the device obtained a less than zero
               exit status code from the script when it was executed.
               The 'message' field below SHOULD indicate both the
               resulting exit status code, as well as capture any
               stdout/stderr messages the script may have produced.
               This update type also indicates that the device has
               abandoned trying to bootstrap off this bootstrap
               server.";
          }
          enum config-warning {
            description
              "Indicates that the device obtained warning messages
               when it committed the initial configuration.  The
               'message' field below SHOULD indicate any warning
               messages that were generated.";
          }
          enum config-error {
            description
              "Indicates that the device obtained error messages
```

```
              when it committed the initial configuration.  The
              'message' field below SHOULD indicate the error
              messages that were generated.  This update type
              also indicates that the device has abandoned trying
              to bootstrap off this bootstrap server.";
        }
        enum post-script-warning {
          description
            "Indicates that the device obtained a greater than
             zero exit status code from the script when it was
             executed.  The 'message' field below SHOULD indicate
             both the resulting exit status code, as well as
             capture any stdout/stderr messages the script may
             have produced.";
        }
        enum post-script-error {
          description
            "Indicates that the device obtained a less than zero
             exit status code from the script when it was executed.
             The 'message' field below SHOULD indicate both the
             resulting exit status code, as well as capture any
             stdout/stderr messages the script may have produced.
             This update type also indicates that the device has
             abandoned trying to bootstrap off this bootstrap
             server.";
        }
        enum bootstrap-complete {
          description
            "Indicates that the device successfully processed the
             all the bootstrapping data and that it is ready to be
             managed.  The 'message' field below MAY contain any
             additional information that the manufacturer thinks
             might be useful.  After sending this update type,
             the device is not expected to access the bootstrap
             server again.";
        }
        enum informational {
          description
            "Indicates any additional information not captured by
             any of the other update type. For instance, a
             message indicating that the device is about to reboot
             after having installed a boot-image could be provided.
             The 'message' field below SHOULD contain information
             that the manufacturer thinks might be useful.";
        }
      }
      mandatory true;
      description
```

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

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

    } // end device

}
<CODE ENDS>
```

8. DHCP Zero Touch Options

   This section defines two DHCP options, one for DHCPv4 and one for
   DHCPv6. These two options are semantically the same, though
   syntactically different.

[8.1](#). **DHCPv4 Zero Touch Option**

   The DHCPv4 Zero Touch Option is used to provision the client with one
   or more URIs for bootstrap servers that can be contacted to attempt
   further configuration.

      DHCPv4 Zero Touch Redirect Option

```
       0                               1
       0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      |    option-code (TBD)    |       option-length     |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      .                                                  .
      .     bootstrap-server-list (variable length)    .
      .                                                  .
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

      o option-code: OPTION_V4_ZEROTOUCH_REDIRECT (TBD)
      o option-length: The option length in octets
      o bootstrap-server-list: A list of servers for the
         client to attempt contacting, in order to obtain
         further bootstrapping data, in the format shown
         in [common-field-encoding].

   DHCPv4 Client Behavior

   Clients MAY request the OPTION_V4_ZEROTOUCH_REDIRECT by including its
   option code in the Parameter Request List (55) in DHCP request
   messages.

   On receipt of a DHCPv4 Reply message which contains the
   OPTION_V4_ZEROTOUCH_REDIRECT, the client performs the following
   steps:

   1.   Check the contents of the DHCPv4 message for at least one valid
        URI. If there is more than one valid URI in the list, a candidate
        list of possible URIs is created.

   2.   Attempt to connect to the one of the URIs in the candidate list.
        The order in which these are processed by the client is
        implementation specific and not defined here.

   3.   If a successful connection to the Zero Touch bootstrap server,
        then the client stops processing entries in the list and proceeds
        according to [Appendix A.3](#), step(3).

4. If the Zero Touch bootstrap server does not respond, provides
    an invalid response, or the transaction otherwise fails, the
    client SHOULD attempt to contact another server from the
    candidate list.

Any invalid URI entries received in the uri-data field are ignored by
the client. If OPTION_V4_ZEROTOUCH_REDIRECT does not contain at
least one valid URI entry in the uri-data field, then the client MUST
discard the option.

DHCPv4 Server Behavior

The DHCPv4 server MAY include a single instance of Option
OPTION_V4_ZEROTOUCH_REDIRECT in DHCP messages it sends. Servers MUST
NOT send more than one instance of the OPTION_V4_ZEROTOUCH_REDIRECT
option.

8.2. DHCPv6 Zero Touch Option

The DHCPv6 Zero Touch Option is used to provision the client with one
or more URIs for bootstrap servers that can be contacted to attempt
further configuration.

   DHCPv6 Zero Touch Redirect Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       option-code (TBD)       |          option-length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.            bootstrap-server-list (variable length)           .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   o option-code: OPTION_V6_ZEROTOUCH_REDIRECT (TBD)
   o option-length: The option length in octets
   o bootstrap-server-list: A list of servers for the client to
     attempt contacting, in order to obtain further bootstrapping
     data, in the format shown in [common-field-encoding].

DHCPv6 Client Behavior

Clients MAY request the OPTION_V6_ZEROTOUCH_REDIRECT option, as
defined in [RFC3315], Sections 17.1.1, 18.1.1, 18.1.3, 18.1.4,
18.1.5, and 22.7.   As a convenience to the reader, we mention here
that the client includes requested option codes in the Option Request
Option.

On receipt of a DHCPv6 reply message which contains the
OPTION_V6_ZEROTOUCH_REDIRECT, the client performs the following
steps:

1. Check the contents of the DHCPv6 message for at least one valid
    URI. If there is more than one valid URI in the list, a
    candidate list of possible URIs is created.

2. Attempt to connect to the one of the URIs in the candidate list.
    The order in which these are processed by the client is
    implementation specific and not defined here.

3. If a successful connection to the Netconf Zero Touch Bootstrap
    server, then the client stops processing entries in the list and
    proceeds according to Appendix A.3, step(3).

4. If the Zero Touch bootstrap server does not respond, provides
    and invalid response or the transaction otherwise fails, the
    client SHOULD attempt to contact another server from the
    candidate list.

Any invalid URI entries received in the uri-data field are ignored by
the client. If OPTION_V6_ZEROTOUCH_REDIRECT does not contain at
least one valid URI entry in the uri-data field, then the client MUST
discard the option.

DHCPv6 Server Behavior

Sections 17.2.2 and 18.2 of [RFC3315] govern server operation
in regard to option assignment. As a convenience to the reader,
we mention here that the server will send a particular option code
only if configured with specific values for that option code and if
the client requested it.

Option OPTION_V6_ZEROTOUCH_REDIRECT is a singleton. Servers MUST NOT
send more than one instance of the OPTION_V6_ZEROTOUCH_REDIRECT
option.

8.3. Common Field Encoding

Both of the DHCPv4 and DHCPv6 options defined in this section encode
a list of bootstrap server URIs. The 'URI' structure is an option
that can contain multiple URIs (see [RFC7227], Section 5.7).

   bootstrap-server-list:

    +-...-+-+-+-+-+-+-+
    | uri-length | URI |
    +-...-+-+-+-+-+-+-+

   o uri-length: variable, in octets.

   o URI: URI of zerotouch bootstrap server, using the HTTPS URI
     scheme defined in Section 2.7.2 of RFC7230.

9. Security Considerations

9.1. Immutable storage for trust anchors

   Devices MUST ensure that all their trust anchor certificates,
   including those for connecting to bootstrap servers and verifying
   ownership vouchers, are protected from external modification.

   It may be necessary to update these certificates over time (e.g., the
   manufacturer wants to delegate trust to a new CA). It is therefore
   expected that devices MAY update these trust anchors when needed
   through a verifiable process, such as a software upgrade using signed
   software images.

9.2. Clock Sensitivity

   The solution in this document relies on TLS certificates, owner
   certificates, and ownership vouchers, all of which require an
   accurate clock in order to be processed correctly (e.g., to test
   validity dates and revocation status). Implementations MUST ensure
   devices have an accurate clock when shipped from manufacturing
   facilities, and take steps to prevent clock tampering.

   If it is not possible to ensure clock accuracy, it is RECOMMENDED
   that implementations disable the aspects of the solution having clock
   sensitivity. In particular, such implementations should assume that
   TLS certificates, ownership vouchers, and owner certificates never
   expire and are not revokable. In real-world terms, this means that
   manufacturers SHOULD only issue a single ownership voucher for the
   lifetime of such devices.

   Implementations SHOULD NOT rely on NTP for time, as NTP is not a
   secure protocol.

9.3. **Blindly authenticating a bootstrap server**

   This document allows a device to blindly authenticate a bootstrap
   server's TLS certificate. It does so to allow for cases where the
   redirect information may be obtained in an unsecured manner, which is
   desirable to support in some cases.

   To compensate for this, this document requires that devices, when
   connected to an untrusted bootstrap server, do not send their IDevID
   certificate for client authentication, and they do not POST any
   progress updates, and they assert that data downloaded from the
   server is signed.

9.4. **Entropy loss over time**

   Section 7.2.7.2 of the IEEE Std 802.1AR-2009 standard says that
   IDevID certificate should never expire (i.e. having the notAfter
   value 99991231235959Z). Given the long-lived nature of these
   certificates, it is paramount to use a strong key length (e.g.,
   512-bit ECC).

9.5. **Serial Numbers**

   This draft uses the device's serial number both in the IDevID
   certificate as well as in the bootstrap server API. 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.

9.6. **Disclosing Information to Untrusted Servers**

   This document enables devices to establish provisional connections to
   untrusted bootstrap servers, in order for the server to provide
   signed data to the device. However, since the server is untrusted,
   it may be under the control of an adversary, and therefore devices
   should be cautious about the data they send in such cases.

   Already this document prevents devices from sending either their
   IDevID certificate or their progress updates to untrusted servers,
   however potentially identifying values (e.g., unique-id, os-name, os-
   version) are still potentially encoded in the HTTP request sent to
   the servers. This information could be used by an adversary to mount
   an attack against the device.

Devices concerned about such disclosures, when connecting to an
untrusted server, should 1) use a 'unique-id' value that is not
easily associated to the type of device it is (e.g., use a key
derivation function over the unique-id value) and 2) be very
selective about which HTTP query parameters are sent (ideally none).

Building on top of the recursive nature of the solution presented in
this document, a server, seeing that a device is following the
recommendation from the previous paragraph, should return signed
redirect information, not signed onboarding information, as would
normally be expected. The redirect information should point the
device back to the same server but, because the signed redirect
information contained the trust anchor certificate for the server,
the device would then establish a trusted connection to the server,
and thereby be able to freely send data to the server, including its
IDevID certificate, unique-id, query parameters, and various progress
updates. Please see Appendix B for more information on this.

9.7. Sequencing Sources of Bootstrapping Data

For devices supporting more than one source for bootstrapping data,
no particular sequencing order has to be observed for security
reasons, as the solution for each source is considered equally
secure. However, from a privacy perspective, it is RECOMMENDED that
devices access local sources before accessing remote sources.

10. IANA Considerations

10.1. The BOOTP Manufacturer Extensions and DHCP Options Registry

IANA is kindly requested to allocate a new option code from the
"BOOTP Manufacturer Extensions and DHCP Options" registry maintained
at http://www.iana.org/assignments/bootp-dhcp-parameters:

        TBD for OPTION_V4_ZEROTOUCH_REDIRECT

And a new option code from the "Dynamic Host Configuration Protocol
for IPv6 (DHCPv6)" registry maintained at
http://www.iana.org/assignments/dhcpv6-parameters:

        TBD for OPTION_V6_ZEROTOUCH_REDIRECT

10.2. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688].
Following the format in [RFC3688], the following registrations are
requested:

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

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

10.3. The YANG Module Names Registry

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

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

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

10.4. RESTCONF Capability URNs

   This document registers several capability identifiers in the
   "RESTCONF Capability URNs" registry per [RFC8040]:

   Index   Capability Identifier

   :os-name   urn:ietf:params:restconf:capability:os-name:1.0
   :os-version   urn:ietf:params:restconf:capability:os-version:1.0
   :circuit-id   urn:ietf:params:restconf:capability:circuit-id:1.0
   :remote-id   urn:ietf:params:restconf:capability:remote-id:1.0
   :nonce   urn:ietf:params:restconf:capability:nonce:1.0

11. Acknowledgements

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

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

12. References

12.1. Normative References

[I-D.ietf-anima-voucher]
          Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,
          "Voucher Profile for Bootstrapping Protocols", draft-ietf-
          anima-voucher-05 (work in progress), August 2017.

[RFC1035] Mockapetris, P., "Domain names - implementation and
          specification", STD 13, RFC 1035, DOI 10.17487/RFC1035,
          November 1987, <https://www.rfc-editor.org/info/rfc1035>.

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

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

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins,
          C., and M. Carney, "Dynamic Host Configuration Protocol
          for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July
          2003, <https://www.rfc-editor.org/info/rfc3315>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
          Housley, R., and W. Polk, "Internet X.509 Public Key
          Infrastructure Certificate and Certificate Revocation List
          (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
          <https://www.rfc-editor.org/info/rfc5280>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
          the Network Configuration Protocol (NETCONF)", RFC 6020,
          DOI 10.17487/RFC6020, October 2010,
          <https://www.rfc-editor.org/info/rfc6020>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
          Verification of Domain-Based Application Service Identity
          within Internet Public Key Infrastructure Using X.509
          (PKIX) Certificates in the Context of Transport Layer
          Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
          2011, <https://www.rfc-editor.org/info/rfc6125>.

   [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms
              (SHA and SHA-based HMAC and HKDF)", RFC 6234,
              DOI 10.17487/RFC6234, May 2011,
              <https://www.rfc-editor.org/info/rfc6234>.

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

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

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

   [RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and
              S. Krishnan, "Guidelines for Creating New DHCPv6 Options",
              BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014,
              <https://www.rfc-editor.org/info/rfc7227>.

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

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

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

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [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>.

12.2. Informative References

[I-D.ietf-netconf-netconf-client-server]
          Watsen, K., Wu, G., and J. Schoenwaelder, "NETCONF Client
          and Server Models", draft-ietf-netconf-netconf-client-
          server-04 (work in progress), July 2017.

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

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

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

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication
          of Named Entities (DANE) Transport Layer Security (TLS)
          Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August
          2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A.,
          Galperin, S., and C. Adams, "X.509 Internet Public Key
          Infrastructure Online Certificate Status Protocol - OCSP",
          RFC 6960, DOI 10.17487/RFC6960, June 2013,
          <https://www.rfc-editor.org/info/rfc6960>.

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

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

Appendix A. Workflow Overview

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

A.1. Enrollment and Ordering Devices

   The following diagram illustrates key interactions that may occur
   from when a prospective owner enrolls in a manufacturer's zero touch
   program to when the manufacturer ships devices for an order placed by
   the prospective owner.

```
                               +-----------+
      +------------+           |Prospective|                  +---+
      |Manufacturer|           |   Owner   |                  |NMS|
      +------------+           +-----------+                  +---+
           |                        |                           |
           |                        |                           |
           |  1. initiate enrollment|                           |
           #<-----------------------|                           |
           #                        |                           |
           #                        |                           |
           #     IDevID trust anchor|                           |
           #---------------------------># set IDevID trust anchor|
           #                          #--------------------------->|
           #                        |                           |
           #     bootstrap server   |                           |
           #     account credentials|                           |
           #---------------------------># set credentials        |
           |                          #--------------------------->|
           |                        |                           |
           |                        |                           |
           |  2. set owner certificate trust anchor             |
           |<--------------------------------------------------|
           |                        |                           |
           |                        |                           |
           |  3. place device order |                           |
           |<--------------------------# model devices          |
           |                          #--------------------------->|
           |                        |                           |
           |  4. ship devices and send|                         |
           |     device identifiers and|                        |
           |     ownership vouchers |                           |
           |---------------------------># set device identifiers |
           |                          #  and ownership vouchers  |
           |                          #--------------------------->|
           |                        |                           |
```
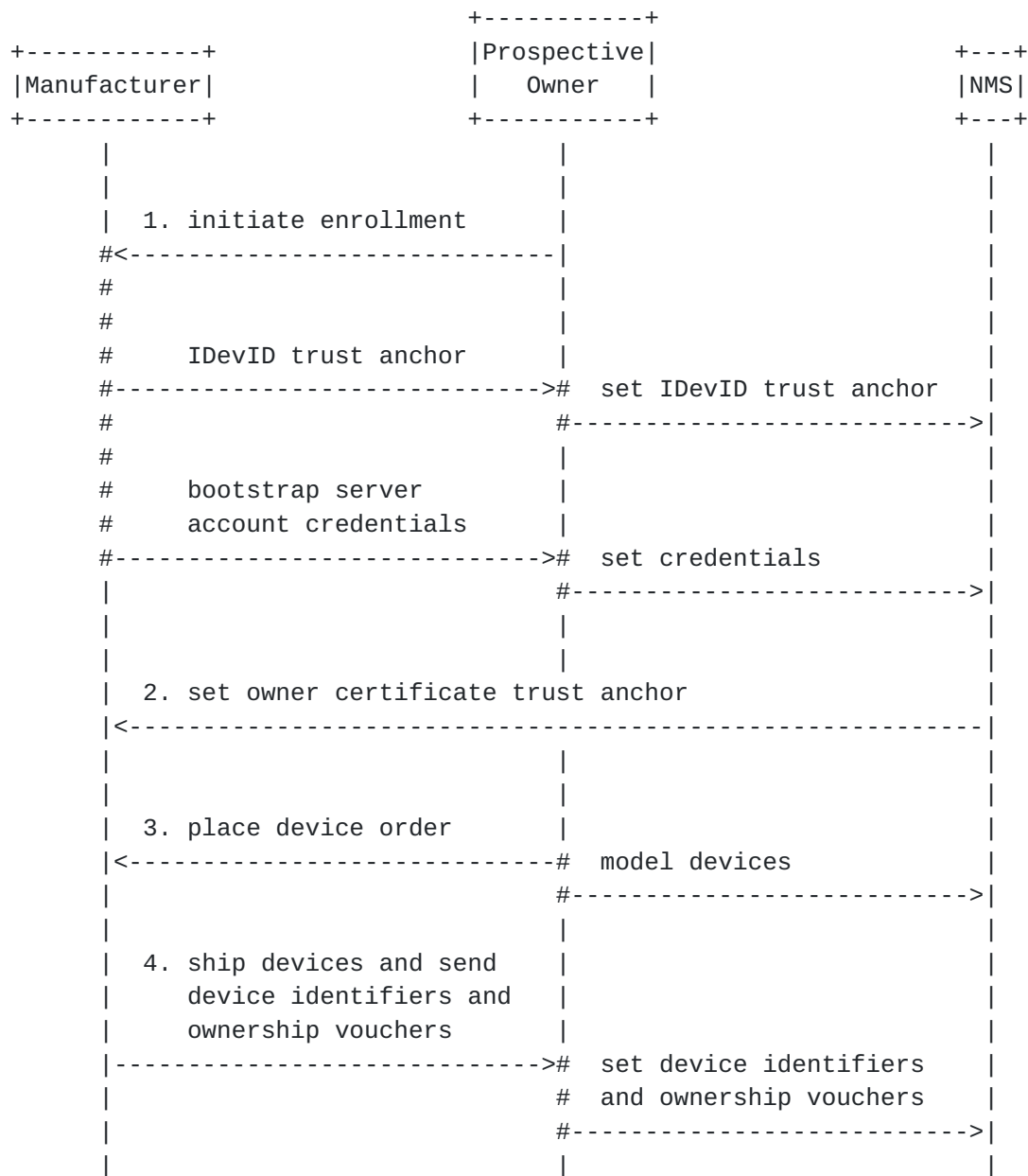
   Each numbered item below corresponds to a numbered item in the
   diagram above.

   1. A prospective owner of a manufacturer's devices, or an existing
       owner that wishes to start using zero touch for future device
       orders, initiates an enrollment process with the manufacturer or
       delegate. This process includes the following:

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

          installed on the prospective owner's NMS so that the NMS can
          subsequently authenticate the devices' IDevID certificates.

     * If the manufacturer hosts an Internet based bootstrap server
        (e.g., a redirect server) such as described in Section 4.4,
        then credentials necessary to configure the bootstrap server
        would be provided to the prospective owner. If the bootstrap
        server is configurable through an API (outside the scope of
        this document), then the credentials might be installed on the
        prospective owner's NMS so that the NMS can subsequently
        configure the manufacturer-hosted bootstrap server directly.

2. If the manufacturer's devices are able to validate signed data
    (Section 5.4), and assuming that the prospective owner's NMS is
    able to prepare and sign the bootstrapping data itself, the
    prospective owner's NMS might set a trust anchor certificate onto
    the manufacturer's bootstrap server, using the credentials
    provided in the previous step. This certificate is the trust
    anchor certificate that the prospective owner would like the
    manufacturer to place into the ownership vouchers it generates,
    thereby enabling devices to trust the owner's owner certificate.
    How this trust anchor certificate is used to enable devices to
    validate signed bootstrapping data is described in Section 5.4.

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

4. When the manufacturer or delegate fulfills the order, shipping
    the devices to their intended locations, they may notify the
    owner of the devices's serial numbers and shipping destinations,
    which the owner may use to stage the network for when the devices
    power on. Additionally, the manufacturer may send one or more
    ownership vouchers, cryptographically assigning ownership of
    those devices to the owner. The owner may set this information
    on their NMS, perhaps binding specific modeled devices to the
    serial numbers and ownership vouchers.

A.2. Owner Stages the Network for Bootstrap

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

```
              +----------+ +------------+
              |Deployment| |Manufacturer| +------+ +------+
              | Specific | |   Hosted   | | Local| | Local| +----------+
        +---+ |Bootstrap | | Bootstrap  | | DNS  | | DHCP | |Removable|
        |NMS| | Server   | |  Server    | |Server| |Server| | Storage |
        +---+ +----------+ +------------+ +------+ +------+ +----------+
          |        |            |            |        |        |
   activate |       |            |            |        |        |
   modeled |        |            |            |        |        |
 1. device |        |            |            |        |        |
 ---------->|        |            |            |        |        |
          | 2. (optional)        |            |        |        |
          |    configure         |            |        |        |
          |    bootstrap         |            |        |        |
          |    server            |            |        |        |
          |------->|             |            |        |        |
          |        |             |            |        |        |
          | 3. (optional) configure           |        |        |
          |    bootstrap server  |            |        |        |
          |--------------------->|            |        |        |
          |        |             |            |        |        |
          |        |             |            |        |        |
          | 4. (optional) configure DNS server|        |        |
          |---------------------------------->|        |        |
          |        |             |            |        |        |
          |        |             |            |        |        |
          | 5. (optional) configure DHCP server        |        |
          |-------------------------------------------->|        |
          |        |             |            |        |        |
          |        |             |            |        |        |
          | 6. (optional) store bootstrapping artifacts on media |
          |----------------------------------------------------->|
          |        |             |            |        |        |
          |        |             |            |        |        |
```
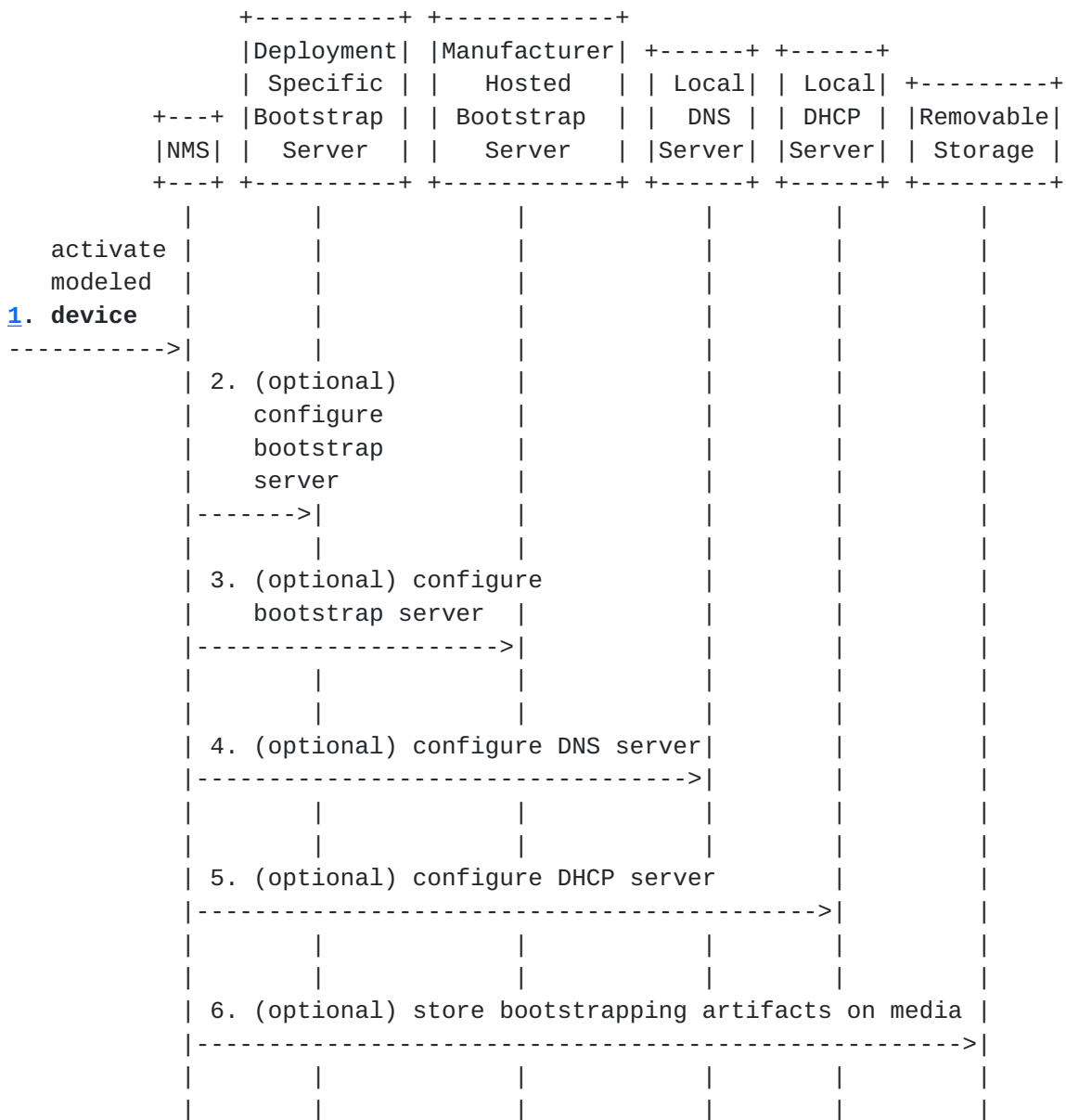
   Each numbered item below corresponds to a numbered item in the
   diagram above.

   1. Having previously modeled the devices, including setting their
       fully operational configurations and associating both device
       serial numbers and ownership vouchers, the owner might "activate"
       one or more modeled devices. That is, the owner tells the NMS to
       perform the steps necessary to prepare for when the real-world
       devices power up and initiate the bootstrapping process. Note
       that, in some deployments, this step might be combined with the
       last step from the previous workflow. Here it is depicted that
       an NMS performs the steps, but they may be performed manually or
       through some other mechanism.

2. If it is desired to use a deployment specific bootstrap server,
    it must be configured to provide the bootstrapping information
    for the specific devices. Configuring the bootstrap server may
    occur via a programmatic API not defined by this document.
    Illustrated here as an external component, the bootstrap server
    may be implemented as an internal component of the NMS itself.

3. If it is desired to use a manufacturer (or delegate) hosted
    bootstrap server, it must be configured to provide the
    bootstrapping information for the specific devices. The
    configuration must be either redirect or onboarding information.
    That is, either the manufacturer hosted bootstrap server will
    redirect the device to another bootstrap server, or provide the
    device with its bootstrapping information itself. The types of
    bootstrapping information the manufacturer hosted bootstrap
    server supports may vary by implementation; some implementations
    may only support redirect information, or only support onboarding
    information, or support both redirect and onboarding information.
    Configuring the bootstrap server may occur via a programmatic API
    not defined by this document.

4. If it is desired to use a DNS server to supply bootstrapping
    information, a DNS server needs to be configured. If multicast
    DNS-SD is desired, then the server must reside on the local
    network, otherwise the DNS server may reside on a remote network.
    Please see Section 4.2 for more information about how to
    configure DNS servers. Configuring the DNS server may occur via
    a programmatic API not defined by this document.

5. If it is desired to use a DHCP server to supply bootstrapping
    data, a DHCP server needs to be configured. The DHCP server may
    be accessed directly or via a DHCP relay. Please see Section 4.3
    for more information about how to configure DHCP servers.
    Configuring the DHCP server may occur via a programmatic API not
    defined by this document.

6. If it is desired to use a removable storage device (e.g., USB
    flash drive) to supply bootstrapping information, the information
    would need to be placed onto it. Please see Section 4.1 for more
    information about how to configure a removable storage device.

A.3. Device Powers On

The following diagram illustrates the sequence of activities that
occur when a device powers on.

```
                                        +----------+
                          +-----------+  |Deployment|
                          | Source of |  | Specific |
   +------+               | Bootstrap |  |Bootstrap |  +---+
   |Device|               |   Data    |  |  Server  |  |NMS|
   +------+               +-----------+  +----------+  +---+
      |                         |             |          |
      |                         |             |          |
      | 1. if zerotouch bootstrap is not      |          |
      |    configured, then exit.             |          |
      |                         |             |          |
      | 2. for each source supported, check   |          |
      |-------------------------------------->|          |
      |                         |             |          |
      | 3. if onboarding-information found,    |          |
      |    initialize self and, only if       |          |
      |    source is a bootstrap server,      |          |
      |    send progress updates              |          |
      |-------------------------------------># |          |
      |                             # webhook  |          |
      |                             #----------------------->|
      |                         |             |          |
      | 4. else if redirect-information found, for        |
      |    each bootstrap server specified, check         |
      |-+---------------------------------------------->| |
      | |                       |             |          |
      | |   if more redirect-information is found, recurse  |
      | |   (not depicted), else if onboarding-information   |
      | |   found, initialize self and post progress updates |
      | +------------------------------------------------># |
      |                                     # webhook |
      |                                     #-------->|
      |
      | 5. retry sources and/or wait for manual provisioning.
      |
```

The interactions in the above diagram are described below.

1. Upon power being applied, the device checks to see if zerotouch
   bootstrapping is configured, such as is the case when running its
   "factory default" configuration. If zerotouch bootstrapping is
   not configured, then the bootstrapping logic exits and none of
   the following interactions occur.

2. For each source of bootstrapping data the device supports,
   preferably in order of closeness to the device (e.g., removable
   storage before Internet based servers), the device checks to see
   if there is any bootstrapping data for it there.

3. If onboarding information is found, the device initializes itself
    accordingly (e.g., installing a boot-image and committing an
    initial configuration). If the source is a bootstrap server, and
    the bootstrap server can be trusted (i.e., TLS-level
    authentication), the device also sends progress updates to the
    bootstrap server.

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

    *   If the bootstrap server supports forwarding device progress
        updates to external systems (e.g., via a webhook), a
        "bootstrap-complete" progress update (Section 7.4) informs the
        external system to know when it can, for instance, initiate a
        connection to the device (assuming it knows the device's
        address and the device was configured to listen for
        connections). To support this further, the bootstrap-complete
        progress update may also relay the device's SSH host keys and/
        or TLS certificates, with which the external system can use to
        authenticate subsequent connections to the device.

    If the device successfully completes the bootstrapping process
    (i.e., zerotouch bootstrapping is no longer configured), it exits
    the bootstrapping logic without considering any additional
    sources of bootstrapping data.

4. Otherwise, if redirect information is found, the device iterates
    through the list of specified bootstrap servers, checking to see
    if there is any bootstrapping data for it on them. If the
    bootstrap server returns more redirect information, then the
    device processes it recursively. Otherwise, if the bootstrap
    server returns onboarding information, the device processes it
    following the description provided in (3) above.

5. After having tried all supported sources of bootstrapping data,
    the device may retry again all the sources and/or provide
    manageability interfaces for manual configuration (e.g., CLI,
    HTTP, NETCONF, etc.). If manual configuration is allowed, and
    such configuration is provided, the device should cease trying to
    obtain bootstrapping data, as the need to do so would no longer
    be present.

Appendix B. Promoting a Connection from Untrusted to Trusted

   The following diagram illustrates a sequence of bootstrapping
   activities that promote an untrusted connection to a bootstrap server
   to a trusted connection to the same bootstrap server.

```
                                          +----------+
                                          |Deployment|
                                          | Specific |
     +------+                             |Bootstrap |
     |Device|                             | Server   |
     +------+                             +----------+
        |                                      |
        | 1. "HTTPS" Request (GET /device=<kdf(unique-id)>)     |
        |--------------------------------------------------->|
        | 2. "HTTPS" Response (signed redirect information)     |
        |<---------------------------------------------------|
        |                                      |
        |                                      |
        | 3. HTTPS Request (GET /device=unique-id?os-name=xyz)   |
        |--------------------------------------------------->|
        | 4. HTTPS Response (unsigned onboarding information     |
        |<---------------------------------------------------|
        |                                      |
```

   The interactions in the above diagram are described below.

   1. The device initiates an untrusted connection to a bootstrap
       server, as is indicated by putting "HTTPS" in doublequotes above.
       Because the device is unable to trust the server, it does not
       send its IDevID certificate, or its raw unique id, or any HTTP
       query parameters it may have. The diagram above uses
       <kdf(unique-id)> to indicate that the device used a key
       derivation function, KDF, over its unique-id to protect the
       device's identity in case the server is owned by an adversary.

   2. The bootstrap server iterates over some internal database of
       devices until finding one for which the key derivation function
       over its unique-id generates a match. In this example, the
       bootstrap server wants the device to have a secure connection to
       it so, rather than sending the device signed onboarding
       information, it sends signed redirect information instead,
       redirecting the device back to it again.

   3. Upon validating the signed redirect information, the device
       establishes a secure connection to the bootstrap server.
       Unbeknownst to the device, it is the same bootstrap server it was
       connected to previously. Because the device is able to

authenticate the bootstrap server, it sends its IDevID
certificate, raw unique-id, and any additional query parameters
it has ("os-name" shown above) to the bootstrap server.

4. This time the bootstrap server returns unsigned onboarding
information to the device.

Appendix C. **Change Log**

C.1. **ID to 00**

o Major structural update; the essence is the same. Most every
section was rewritten to some degree.

o Added a Use Cases section

o Added diagrams for "Actors and Roles" and "NMS Precondition"
sections, and greatly improved the "Device Boot Sequence" diagram

o Removed support for physical presence or any ability for
configlets to not be signed.

o Defined the Zero Touch Information DHCP option

o Added an ability for devices to also download images from
configuration servers

o Added an ability for configlets to be encrypted

o Now configuration servers only have to support HTTP/S - no other
schemes possible

C.2. **00 to 01**

o Added boot-image and validate-owner annotations to the "Actors and
Roles" diagram.

o Fixed 2nd paragraph in section 7.1 to reflect current use of
anyxml.

o Added encrypted and signed-encrypted examples

o Replaced YANG module with XSD schema

o Added IANA request for the Zero Touch Information DHCP Option

o Added IANA request for media types for boot-image and
configuration

C.3. **01 to 02**

   o Replaced the need for a configuration signer with the ability for
      each NMS to be able to sign its own configurations, using
      manufacturer signed ownership vouchers and owner certificates.

   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

C.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.

C.5. **03 to 04**

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

   o Added many terms to Terminology section.

   o Added all new "Guiding Principles" section.

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

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

C.6. **04 to 05**

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

   o Created new section for information types

   o Added support for DNS servers

   o Now allows provisional TLS connections

   o Bootstrapping data now supports scripts

o Device Details section overhauled

o Security Considerations expanded

o Filled in enumerations for notification types

C.7. 05 to 06

o Minor update

o Added many Normative and Informative references.

o Added new section Other Considerations.

C.8. 06 to 07

o Minor update

o Added an Editorial Note section for RFC Editor.

o Updated the IANA Considerations section.

C.9. 07 to 08

o Minor update

o Updated to reflect review from Michael Richardson.

C.10. 08 to 09

o Added in missing "Signature" artifact example.

o Added recommendation for manufacturers to use interoperable
   formats and file naming conventions for removable storage devices.

o Added configuration-handling leaf to guide if config should be
   merged, replaced, or processed like an edit-config/yang-patch
   document.

o Added a pre-configuration script, in addition to the post-
   configuration script from -05 (issue #15).

C.11. 09 to 10

o Factored ownership vocher and voucher revocation to a separate
   document: draft-kwatsen-netconf-voucher. (issue #11)

o Removed <configuration-handling> options 'edit-config' and yang-
   patch'. (issue #12)

o Defined how a signature over signed-data returned from a bootstrap
   server is processed. (issue #13)

o Added recommendation for removable storage devices to use open/
   standard file systems when possible. (issue #14)

o Replaced notifications "script-[warning/error]" with "[pre/post]-
   script-[warning/error]". (goes with issue #15)

o switched owner-certificate to be encoded using the pkcs#7 format.
   (issue #16)

o Replaced md5/sha1 with sha256 inside a choice statement, for
   future extensibility. (issue #17)

o A ton of editorial changes, as I went thru the entire draft with a
   fine-toothed comb.

C.12. 10 to 11

o fixed yang validation issues found by IETFYANGPageCompilation.
   note: these issues were NOT found by pyang --ietf or by the
   submission-time validator...

o fixed a typo in the yang module, someone the config false
   statement was removed.

C.13. 11 to 12

o fixed typo that prevented Appendix B from loading the examples
   correctly.

o fixed more yang validation issues found by
   IETFYANGPageCompilation. note: again, these issues were NOT found
   by pyang --ietf or by the submission-time validator...

o updated a few of the notification enumerations to be more
   consistent with the other enumerations (following the warning/
   error pattern).

o updated the information-type artifact to state how it's encoded,
   matching the language that was in Appendix B.

   o defined a standalone artifact to encode the old information-type
      into a PKCS#7 structure.

   o standalone information artifact hardcodes JSON encoding (to match
      the voucher draft).

   o combined the information and signature PKCS#7 structures into a
      single PKCS#7 structure.

   o moved the certificate-revocations into the owner-certificate's
      PKCS#7 structure.

   o eliminated support for voucher-revocations, to reflect the
      voucher-draft's switch from revocations to renewals.

   o Renamed "bootstrap information" to "onboarding information".

   o Rewrote DHCP sections to address the packet-size limitation issue,
      as discussed in Chicago.

   o Added Ian as an author for his text-contributions to the DHCP
      sections.

   o Removed the Guiding Principles section.

   o Renamed action 'notification' to 'update-progress' and, likewise
      'notification-type' to 'update-type'.

   o Updated examples to use "base64encodedvalue==" for binary values.

   o Greatly simplified the 'Artifact Groupings' section, and moved it
      as a subsection to the 'Artifacts' section.

   o Moved the 'Workflow Overview' section to the Appendix.

   o Renamed 'bootstrap information' to 'update information'.

   o Removed 'Other Considerations' section.

   o Tons of editorial updates.

C.17. 15 to 16

o tweaked language to refer to "initial state" rather than "factory
   default configuration", so as accommodate white-box scenarios.

o added a paragraph to Intro regarding how the solution primarily
   regards physical machines, but could be extended to VMs by a
   future document.

o added a pointer to the Workflow Overview section (recently moved
   to the Appendix) to the Intro.

o added a note that, in order to simplify the verification process,
   the "Zerotouch Information" PKCS#7 structure MUST also contain the
   signing X.509 certificate.

o noted that the owner certificate's must either have no Key Usage
   or the Key Usage must set the "digitalSignature" bit.

o noted that the owner certificate's subject and subjectAltName
   values are not constrained.

o moved/consolidated some text from the Artifacts section down to
   the Device Details section.

o tightened up some ambiguous language, for instance, by referring
   to specific leaf names in the Voucher artifact.

o reverted a previously overzealous s/unique-id/serial-number/
   change.

o modified language for when ZTP runs from when factory-default
   config is running to when ZTP is configured, which the factory-
   defaults should set .

C.18. 16 to 17

o Added an example for how to promote an untrusted connection to a
   trusted connection.

o Added a "query parameters" section defining some parameters
   enabling scenarios raised in last call.

o Added a "Disclosing Information to Untrusted Servers" section to
   the Security Considerations.

Authors' Addresses

    Kent Watsen
    Juniper Networks

    EMail: kwatsen@juniper.net

    Mikael Abrahamsson
    T-Systems

    EMail: mikael.abrahamsson@t-systems.se

    Ian Farrer
    Deutsche Telekom AG

    EMail: ian.farrer@telekom.de