             Zero Touch Provisioning for Networking Devices
                    draft-ietf-netconf-zerotouch-26

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

   o "TBD1" --> the assigned value for id-ct-zerotouchInformationXML

   o "TBD2" --> the assigned value for id-ct-zerotouchInformationJSON

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

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

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

   o "2018-12-20" --> the publication date of this draft

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

   o [Appendix D](#). Change Log

Status of This Memo

   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 June 23, 2019.

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

1.1. Use Cases

   o Device connecting to a remotely administered network

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

   o Device connecting to a locally administered network

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

just as it would for the remotely administered network use-
case.

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

1.2. Terminology

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

Artifact: The term "artifact" is used throughout to represent any of
the three artifacts defined in Section 3 (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 zero touch information, owner
certificate, and ownership voucher, as described in Section 3.

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

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

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

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

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

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

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

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

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

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

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

Signed Data: The term "signed data" is used throughout to mean zero
   touch 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
   zero touch information that has not been signed.

Zero Touch Information: The term "zero touch information" is used
   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 Diagrams

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

2. Types of Zero Touch Information

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

2.1. Redirect Information

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

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

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

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

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

Untrusted redirect information is useful for directing a device to a
bootstrap server where signed data has been staged for it to obtain.

Note that, when the redirect information is untrusted, devices
discard any potentially included trust anchor certificates.

How devices process redirect information is described in Section 5.5.

2.2. Onboarding Information

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

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

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

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

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

How devices process onboarding information is described in
Section 5.6.

3. Artifacts

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

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

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

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

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

When the zero touch information artifact is signed and encrypted, as
it might be when communicated over untrusted channels and privacy is

important, the CMS structure's top-most content type MUST be the OID id-envelopedData (1.2.840.113549.1.7.3). Furthermore, the encryptedContentInfo's content type MUST be the OID id-signedData (1.2.840.113549.1.7.2), whose eContentType MUST be one of the OIDs described in Section 10.3 (i.e., id-ct-zerotouchInformationXML or id-ct-zerotouchInformationJSON), or the OID id-data (1.2.840.113549.1.7.1). When the OID id-data is used, the encoding (JSON, XML, etc.) SHOULD be communicated externally. In either case, the associated content or eContent is an octet string containing "zerotouch-information" data in the expected encoding.

3.2. Owner Certificate

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

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

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

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

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

When unencrypted, the owner certificate artifact's CMS structure's top-most content type MUST be the OID id-signedData

(1.2.840.113549.1.7.2). The inner SignedData structure is the
degenerate form, whereby there are no signers, that is commonly used
to disseminate certificates and revocation objects.

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

3.3. Ownership Voucher

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

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

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

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

3.4. Artifact Encryption

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

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

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

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

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

3.5. Artifact Groupings

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

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

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

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

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

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

4. Sources of Bootstrapping Data

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

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

4.1. Removable Storage

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

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

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

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

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

Artifact to File Mapping:

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

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

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

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

4.2. DNS Server

A DNS server MAY be used as a source of 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.

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

4.2.1. DNS Queries

Devices claiming to support DNS as a source of bootstrapping data MUST first query for device-specific DNS records using DNS-SD [RFC6763] and, only if doing so does not result in a successful bootstrap, then query for device-independent records using traditional service discovery [RFC2782]. Furthermore, when issuing the device-specific and device-independent queries, devices MUST first query using multicast DNS [RFC6762] and, only if doing so does not result in a successful bootstrap, then query again using unicast DNS [RFC1035][RFC7766], assuming the address of a DNS server is known.

When querying for device-specific DNS records, devices SHOULD immediately query for their instance-specific record, without first querying for PTR records (Section 4.1 of [RFC6763]). Device MUST use their serial number (i.e., the same value as in the device identity certificate) for the "<instance>" portion of the service instance name (Section 4.1.1 of [RFC6763]). Device MUST only query for TXT records, in order to access the three bootstrapping artifacts defined in Section 3.

When querying for device-independent DNS records, devices MUST only query for SRV records. Multiple SRV records, each specifying an address, port, weight, and priority [RFC2782] is comparable to an unsigned redirect information's list of bootstrap servers. Note that a device-independent response is only able to encode unsigned data, since signed data necessitates the use of a device-specific ownership voucher. Exclusive use of SRV records maximally leverages existing DNS standards for what is likely to be a common case when using a DNS server as a source of bootstrapping data.

By example, assuming a device's serial number is "<serial number>", and that the domain discovered per Section 11 of [RFC6763] is "example.com", the device may issue the following sequence of DNS queries:

```
TXT in <serial number>._zerotouch._tcp.local.
TXT in <serial number>._zerotouch._tcp.example.com.
```

      SRV in _zerotouch._tcp.local.
      SRV in _zerotouch._tcp.example.com.

   This document registers the service name "zerotouch" in Section 10.6.

4.2.2. DNS Response for Device-Specific Queries

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

   Artifact to TXT Record Mapping:

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

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

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

   Devices MUST ignore any other keys that may be returned.

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

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

     <sn>._zerotouch._tcp.example.com. 3600 IN TXT "zi=<binary data>"
     <sn>._zerotouch._tcp.example.com. 3600 IN TXT "oc=<binary data>"
     <sn>._zerotouch._tcp.example.com. 3600 IN TXT "ov=<binary data>"

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

4.2.3. DNS Response for Device-Independent Queries

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

   Artifact to SRV Record Mapping:

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

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

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

   Following is an example of a device-independent response, as it might
   be presented by a user-agent, containing (effectively) unsigned
   redirect information to four bootstrap servers:

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

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

4.2.4. Size of Signed Data

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

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

   Given the expected size of the TXT records, it is unlikely that
   signed data will fit into a UDP-based DNS packet, even with the
   EDNS(0) Extensions [RFC6891] enabled. Depending on content, signed
   data may also not fit into a multicast DNS packet, which bounds the
   size of the TXT records to 8900 bytes, per Section 6.1 in [RFC6763].
   Thus it is expected that DNS Transport over TCP [RFC7766] will be
   required in order to return signed data.

4.3. DHCP Server

   A DHCP server MAY be used as a source of 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 very
   limited in the amount of data that can be conveyed, to the extent
   that signed data cannot be communicated. This means that only
   unsigned redirect information can be conveyed via DHCP.

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

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

   Artifact to DHCP Option Fields Mapping:

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

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

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

4.4. Bootstrap Server

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

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

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

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

From an artifact perspective, since a bootstrap server presents data
conforming to a YANG data model, the bootstrapping artifacts need to
be mapped to YANG nodes. The three artifacts defined in Section 3
are mapped to "output" nodes of the "get-bootstrapping-data" RPC
defined in Section 7.3 below.

Artifact to Bootstrap Server Mapping:

   Zero Touch Information: Mapped to the "zerotouch-information"
      leaf in the output of the "get-bootstrapping-data" RPC.

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

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

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

5. Device Details

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

5.1. Initial State

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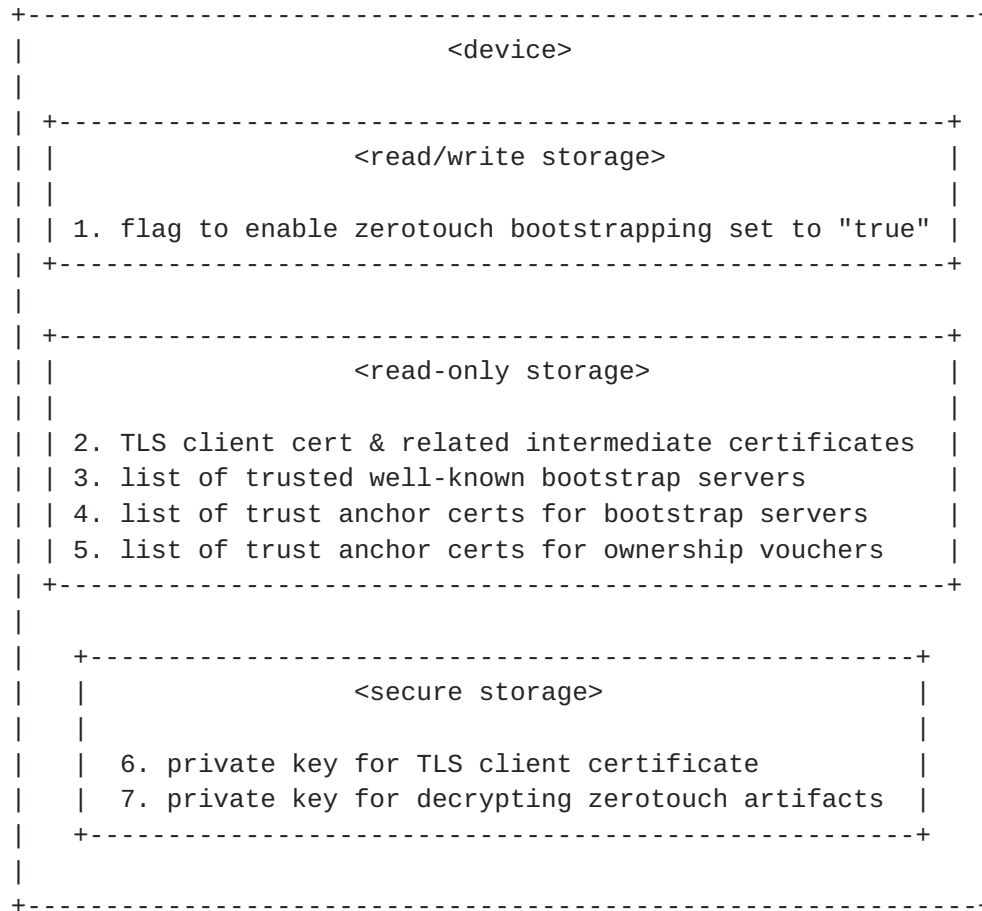
   Each numbered item below corresponds to a numbered item in the
   diagram above.

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

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

certificate, devices MAY, alternatively or in addition to, identify and authenticate themselves to the bootstrap server using an HTTP authentication scheme, as allowed by Section 2.5 in [RFC8040]; however, this document does not define a mechanism for operator input enabling, for example, the entering of a password.

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

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

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

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

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

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

5.2. Boot Sequence

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

```
     Power On
        |
        v                              No
  1. 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 config
        |
        | No
        v
  4. Loop back to Step 1.
```
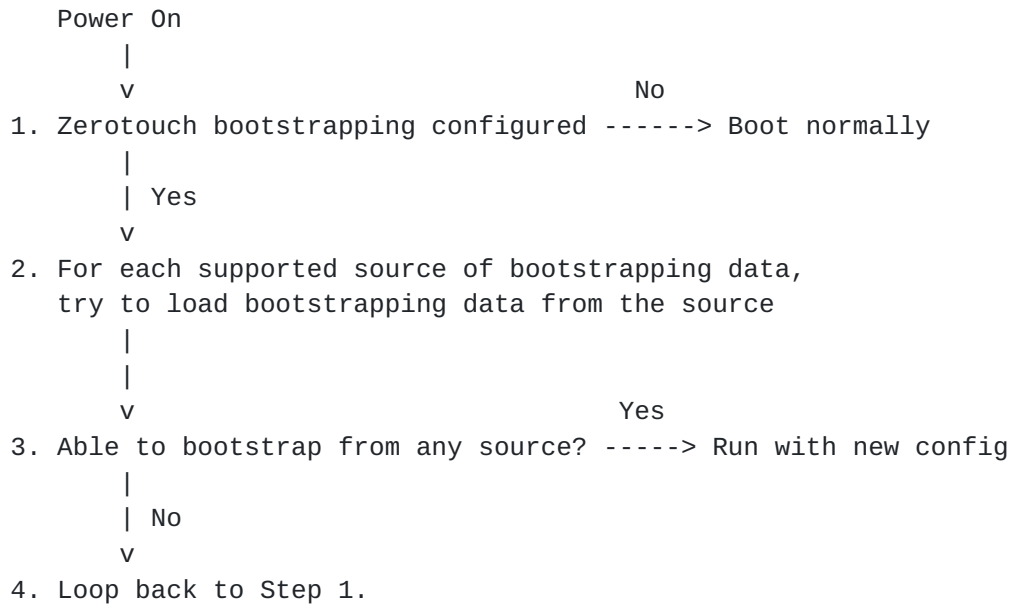

    Note: At any time, the device MAY be configured via an alternate
          provisioning mechanism (e.g., CLI).

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

   1. When the device powers on, it first checks to see if zerotouch
       bootstrapping is configured, as is expected to be the case for
       the device's preconfigured initial 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 MUST loop back through the list of
       bootstrapping sources again.

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

5.3. Processing a Source of Bootstrapping Data

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

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

```
                           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.

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

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

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

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

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

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

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

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

5.4. Validating Signed Data

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

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

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

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

Finally the device MUST verify the zero touch information artifact
was signed by the validated owner certificate.

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

5.5. Processing Redirect Information

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

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

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

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

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

5.6. Processing Onboarding Information

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

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

   When the onboarding information is obtained from a trusted bootstrap
   server, the device MUST send the "bootstrap-initiated" progress
   report, and send either a terminating "boot-image-installed-
   rebooting", "bootstrap-complete", or error specific progress report.

If the bootstrap server's "get-bootstrapping-data" RPC-reply's
"reporting-level" node is set to "verbose", the device MUST
additionally send all appropriate non-terminating progress reports
(e.g., initiated, warning, complete, etc.). Regardless of the
reporting-level indicated by the bootstrap server, the device MAY
send progress reports beyond the mandatory ones specified for the
given reporting level.

When the onboarding information is obtained from an untrusted
bootstrap server, the device MUST NOT send any progress reports to
the bootstrap server.

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

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

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

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

If boot image criteria are specified, the device MUST first determine
if the boot image it is running satisfies the specified boot image
criteria. If the device is already running the specified boot image,
then it skips the remainder of this step. If the device is not
running the specified boot image, then it MUST download, verify, and
install, in that order, the specified boot image, and then reboot.
If connected to a trusted bootstrap server, the device MAY try to
send a "boot-image-mismatch" progress report. To download the boot

image, the device MUST only use the URIs supplied by the onboarding
information. To verify the boot image, the device MUST either use
one of the verification fingerprints supplied by the onboarding
information, or use a cryptographic signature embedded into the boot
image itself using a mechanism not described by this document.
Before rebooting, if connected to a trusted bootstrap server, the
device MUST try to send a "boot-image-installed-rebooting" progress
report. Upon rebooting, the bootstrapping process runs again, which
will eventually come to this step again, but then the device will be
running the specified boot image, and thus will move to processing
the next step. If an error occurs at any step while the device is
connected to a trusted bootstrap server (i.e., before the reboot),
the device MUST try to send a "boot-image-error" progress report
before exiting.

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

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

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

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

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

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

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

Implementation Notes:

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

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

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

   * Most steps are atomic. For example, the processing configuration is specified as atomic above, and the processing scripts is similarly specified as atomic in the "ietf-zerotouch-information" YANG module.

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

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

6. The Zero Touch Information Data Model

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. This data model uses the "yang-data" extension statement defined in [I-D.ietf-netmod-yang-data-ext]. Examples illustrating this data model are provided in Section 6.2.

6.1. Data Model Overview

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

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

6.2. Example Usage

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

```
{
  "ietf-zerotouch-information:redirect-information" : {
    "bootstrap-server" : [
      {
        "address" : "phs1.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      },
      {
        "address" : "phs2.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      },
      {
        "address" : "phs3.example.com",
        "port" : 8443,
        "trust-anchor" : "base64encodedvalue=="
      }
    ]
  }
}
```

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

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

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

6.3. YANG Module

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

This module uses data types defined in [RFC5280], [RFC5652], [RFC6234], and [RFC6991], an extension statement from [I-D.ietf-netmod-yang-data-ext], and an encoding defined in [ITU.X690.2015].

```
<CODE BEGINS> file "ietf-zerotouch-information@2018-12-20.yang"
module ietf-zerotouch-information {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-zerotouch-information";
  prefix zti;

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types";
  }
  import ietf-yang-data-ext {
    prefix yd;
    reference "I-D.ietf-netmod-yang-data-ext: YANG Data Extensions";
  }

  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 in RFC XXXX: Zero Touch
    Provisioning for Networking Devices.

    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.
```

```
  revision 2018-12-20 {
    description
      "Initial version";
    reference
      "RFC XXXX: Zero Touch Provisioning for Networking Devices";
  }

  // identities

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

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

  // typedefs

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

```
      }

      // yang-data

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

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

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

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

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

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

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

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

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

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

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

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

7. The Zero Touch Bootstrap Server API

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

7.1. API Overview

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

```
module: ietf-zerotouch-bootstrap-server

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

7.2. Example Usage

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

The following example illustrates a device using the API to fetch its
bootstrapping data from a untrusted bootstrap server. In this
example, the device sends the "untrusted-connection" input parameter
and receives signed data in the response.

REQUEST

['\' line wrapping added for formatting only]

POST /restconf/operations/ietf-zerotouch-bootstrap-server:get-boot\
strapping-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<input
 xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
</input>

RESPONSE

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

The following example illustrates a device using the API to fetch its
bootstrapping data from a trusted bootstrap server. In this example,
the device sends addition input parameters to the bootstrap server,
which it may use when formulating its response to the device.

REQUEST

['\' line wrapping added for formatting only]

POST /restconf/operations/ietf-zerotouch-bootstrap-server:get-boot\
strapping-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<input
 xmlns="urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <hw-model>model-x</hw-model>
  <os-name>vendor-os</os-name>
  <os-version>17.3R2.1</os-version>
  <nonce>base64encodedvalue==</nonce>
</input>

RESPONSE

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

The following example illustrates a device using the API to post a
progress report to a bootstrap server. Illustrated below is the
"bootstrap-complete" message, but the device may send other progress
reports to the server while bootstrapping. In this example, the
device is sending both its SSH host keys and a TLS server
certificate, which the bootstrap server may, for example, pass to an
NMS, as discussed in Appendix C.3.

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

POST /restconf/operations/ietf-zerotouch-bootstrap-server:report-\
progress HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+xml

<input xmlns=
  "urn:ietf:params:xml:ns:yang:ietf-zerotouch-bootstrap-server">
  <progress-type>bootstrap-complete</progress-type>
  <message>example message</message>
  <ssh-host-keys>
    <ssh-host-key>
      <algorithm>ssh-rsa</algorithm>
      <key-data>base64encodedvalue==</key-data>
    </ssh-host-key>
    <ssh-host-key>
      <algorithm>rsa-sha2-256</algorithm>
      <key-data>base64encodedvalue==</key-data>
    </ssh-host-key>
  </ssh-host-keys>
  <trust-anchor-certs>
    <trust-anchor-cert>base64encodedvalue==</trust-anchor-cert>
  </trust-anchor-certs>
</input>

RESPONSE
--------

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

7.3. YANG Module

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

This module uses data types defined in [RFC4253], [RFC5652],
[RFC5280], [RFC6960], and [RFC8366], uses an encoding defined in
[ITU.X690.2015], and makes a reference to [RFC4250] and [RFC6187].

```
<CODE BEGINS> file "ietf-zerotouch-bootstrap-server@2018-12-20.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 Networking
      Devices.

      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.

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

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

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

    revision 2018-12-20 {
      description
        "Initial version";
      reference
        "RFC XXXX: Zero Touch Provisioning for Networking Devices";
    }

    // typedefs

    typedef cms {
      type binary;
      description
        "A CMS structure, as specified in RFC 5652, encoded using
```

```
         ASN.1 distinguished encoding rules (DER), as specified in
         ITU-T X.690.";
       reference
         "RFC 5652:
            Cryptographic Message Syntax (CMS)
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
     }

     // RPCs

     rpc get-bootstrapping-data {
       description
         "This RPC enables a device, as identified by the RESTCONF
          username, to obtain bootstrapping data that has been made
          available for it.";
       input {
         leaf untrusted-connection {
           type empty;
           description
             "This optional input parameter enables a device to
              communicate to the bootstrap server that it is unable to
              authenticate the bootstrap server's TLS certificate.  In
              such circumstances, the device likely does not send any
              of the other input parameters, except for the 'nonce'
              parameter.  Upon receiving this input parameter, the
              bootstrap server should only return unsigned redirect
              information or signed data of any type.";
         }
         leaf hw-model {
           type string;
           description
             "This optional input parameter enables a device to
              communicate to the bootstrap server its vendor specific
              hardware model number.  This parameter may be needed,
              for instance, when a device's IDevID certificate does
              not include the 'hardwareModelName' value in its
              subjectAltName field, as is allowed by 802.1AR-2009.";
           reference
             "IEEE 802.1AR-2009: IEEE Standard for Local and
                 metropolitan area networks - Secure Device Identity";
         }
         leaf os-name {
           type string;
           description
```

```
            "This optional input parameter enables a device to
             communicate to the bootstrap server the name of its
             operating system.  This parameter may be useful if
             the device, as identified by its serial number, can
             run more than one type of operating system (e.g.,
             on a white-box system.";
        }
        leaf os-version {
          type string;
          description
            "This optional input parameter enables a device to
             communicate to the bootstrap server the version of its
             operating system.  This parameter may be used by a
             bootstrap server to return an operating system specific
             response to the device, thus negating the need for a
             potentially expensive boot-image update.";
        }
        leaf nonce {
          type binary {
            length "8..32";
          }
          description
            "This optional input parameter enables a device to
             communicate to the bootstrap server a nonce value.
             This may be especially useful for devices lacking
             an accurate clock, as then the bootstrap server
             can dynamically obtain from the manufacturer a
             voucher with the nonce value in it, as described
             in RFC 8366.";
          reference
            "RFC 8366:
               A Voucher Artifact for Bootstrapping Protocols";
        }
      }
      output {
        leaf reporting-level {
          type enumeration {
            enum standard {
              description
                "Send just the progress reports required by RFC XXXX.";
            }
            enum verbose {
              description
                "Send additional progress reports that might help
                 troubleshooting an SZTP bootstrapping issue.";
            }
          }
          default standard;
```

```
          description
            "Specifies the reporting level for progress reports the
             bootstrap server would like to receive when processing
             onboarding information.  Progress reports are not sent
             when processing redirect information, or when the
             bootstrap server is untrusted (e.g., device sent the
             '<untrusted-connection>' input parameter).";
        }
        leaf zerotouch-information {
          type cms;
          mandatory true;
          description
            "A zero touch information artifact, as described in
             Section 3.1 of RFC XXXX.";
          reference
            "RFC XXXX:
                Zero Touch Provisioning for Networking Devices";
        }
        leaf owner-certificate {
          type cms;
          must '../ownership-voucher' {
            description
              "An ownership voucher must be present whenever an owner
               certificate is presented.";
          }
          description
            "An owner certificate artifact, as described in Section
             3.2 of RFC XXXX.  This leaf is optional because it is
             only needed when the zero touch information artifact
             is signed.";
          reference
            "RFC XXXX:
                Zero Touch Provisioning for Networking Devices";
        }
        leaf ownership-voucher {
          type cms;
          must '../owner-certificate' {
            description
              "An owner certificate must be present whenever an
               ownership voucher is presented.";
          }
          description
            "An ownership voucher artifact, as described by Section
             3.3 of RFC XXXX.  This leaf is optional because it is
             only needed when the zero touch information artifact
             is signed.";
          reference
            "RFC XXXX:
```

```
                  Zero Touch Provisioning for Networking Devices";
        }
      }
    }

    rpc report-progress {
      description
        "This RPC enables a device, as identified by the RESTCONF
         username, to report its bootstrapping progress to the
         bootstrap server.  This RPC is expected to be used when
         the device obtains onboarding-information from a trusted
         bootstap server.";
      input {
        leaf progress-type {
          type enumeration {
            enum "bootstrap-initiated" {
              description
                "Indicates that the device just used the
                 'get-bootstrapping-data' RPC.  The 'message' node
                 below MAY contain any additional information that
                 the manufacturer thinks might be useful.";
            }
            enum "parsing-initiated" {
              description
                "Indicates that the device is about to start parsing
                 the onboarding information.  This progress type is
                 only for when parsing is implemented as a distinct
                 step.";
            }
            enum "parsing-warning" {
              description
                "Indicates that the device had a non-fatal error when
                 parsing the response from the bootstrap server.  The
                 'message' node below SHOULD indicate the specific
                 warning that occurred.";
            }
            enum "parsing-error" {
              description
                "Indicates that the device encountered a fatal error
                 when parsing the response from the bootstrap server.
                 For instance, this could be due to malformed encoding,
                 the device expecting signed data when only unsigned
                 data is provided, the ownership voucher not listing
                 the device's serial number, or because the signature
                 didn't match.  The 'message' node below SHOULD
                 indicate the specific error.  This progress type
                 also indicates that the device has abandoned trying
                 to bootstrap off this bootstrap server.";
```

```
            }
            enum "parsing-complete" {
              description
                "Indicates that the device successfully completed
                 parsing the onboarding information.  This progress
                 type is only for when parsing is implemented as a
                 distinct step.";
            }
            enum "boot-image-initiated" {
              description
                "Indicates that the device is about to start
                 processing the boot-image information.";
            }
            enum "boot-image-warning" {
              description
                "Indicates that the device encountered a non-fatal
                 error condition when trying to install a boot-image.
                 A possible reason might include a need to reformat a
                 partition causing loss of data.  The 'message' node
                 below SHOULD indicate any warning messages that were
                 generated.";
            }
            enum "boot-image-error" {
              description
                "Indicates that the device encountered an error when
                 trying to install a boot-image, which could be for
                 reasons such as a file server being unreachable,
                 file not found, signature mismatch, etc.  The
                 'message' node SHOULD indicate the specific error
                 that occurred.  This progress type also indicates
                 that the device has abandoned trying to bootstrap
                 off this bootstrap server.";
            }
            enum "boot-image-mismatch" {
              description
                "Indicates that the device that has determined that
                 it is not running the correct boot image.  This
                 message SHOULD precipitate trying to download
                 a boot image.";
            }
            enum "boot-image-installed-rebooting" {
              description
                "Indicates that the device successfully installed
                 a new boot image and is about to reboot.  After
                 sending this progress type, the device is not
                 expected to access the bootstrap server again
                 for this bootrapping attempt.  The device may
                 access this bootstrap server after rebooting
```

```
                        and restarting the zerotouch bootstrapping
                        process.";
                    }
                    enum "boot-image-complete" {
                      description
                        "Indicates that the device believes that it is
                         running the correct boot-image.";
                    }
                    enum "pre-script-initiated" {
                      description
                        "Indicates that the device is about to execute the
                         'pre-configuration-script'.";
                    }
                    enum "pre-script-warning" {
                      description
                        "Indicates that the device obtained a warning from the
                         'pre-configuration-script' when it was executed.  The
                         'message' node below SHOULD capture any output the
                         script produces.";
                    }
                    enum "pre-script-error" {
                      description
                        "Indicates that the device obtained an error from the
                         'pre-configuration-script' when it was executed. The
                         'message' node below SHOULD capture any output the
                         script produces.  This progress type also indicates
                         that the device has abandoned trying to bootstrap
                         off this bootstrap server.";
                    }
                    enum "pre-script-complete" {
                      description
                        "Indicates that the device successfully executed the
                         'pre-configuration-script'.";
                    }
                    enum "config-initiated" {
                      description
                        "Indicates that the device is about to commit the
                         initial configuration.";
                    }
                    enum "config-warning" {
                      description
                        "Indicates that the device obtained warning messages
                         when it committed the initial configuration.  The
                         'message' node below SHOULD indicate any warning
                         messages that were generated.";
                    }
                    enum "config-error" {
                      description
```

```
                    "Indicates that the device obtained error messages
                     when it committed the initial configuration.  The
                     'message' node below SHOULD indicate the error
                     messages that were generated.  This progress type
                     also indicates that the device has abandoned trying
                     to bootstrap off this bootstrap server.";
                }
                enum "config-complete" {
                  description
                    "Indicates that the device successfully committed
                     the initial configuration.";
                }
                enum "post-script-initiated" {
                  description
                    "Indicates that the device is about to execute the
                     'post-configuration-script'.";
                }
                enum "post-script-warning" {
                  description
                    "Indicates that the device obtained a warning from the
                     'post-configuration-script' when it was executed. The
                     'message' node below SHOULD capture any output the
                     script produces.";
                }
                enum "post-script-error" {
                  description
                    "Indicates that the device obtained an error from the
                     'post-configuration-script' when it was executed. The
                     'message' node below SHOULD capture any output the
                     script produces.  This progress type also indicates
                     that the device has abandoned trying to bootstrap
                     off this bootstrap server.";
                }
                enum "post-script-complete" {
                  description
                    "Indicates that the device successfully executed the
                     'post-configuration-script'.";
                }
                enum "bootstrap-warning" {
                 description
                    "Indicates that a warning condition occurred for which
                     there no other 'progress-type' enumeration is deemed
                     suitable.  The 'message' node below SHOULD describe
                     the warning.";
                }
                enum "bootstrap-error" {
                 description
                    "Indicates that an error condition occurred for which
```

```
                there no other 'progress-type' enumeration is deemed
                suitable.  The 'message' node below SHOULD describe
                the error.  This progress type also indicates that
                the device has abandoned trying to bootstrap off
                this bootstrap server.";
          }
          enum "bootstrap-complete" {
            description
              "Indicates that the device successfully processed
               all 'onboarding-information' provided, and that it
               is ready to be managed.  The 'message' node below
               MAY contain any additional information that the
               manufacturer thinks might be useful.  After sending
               this progress type, the device is not expected to
               access the bootstrap server again.";
          }
          enum "informational" {
            description
              "Indicates any additional information not captured
               by any of the other progress types. For instance,
               a message indicating that the device is about to
               reboot after having installed a boot-image could
               be provided.  The 'message' node below SHOULD
               contain information that the manufacturer thinks
               might be useful.";
          }
        }
        mandatory true;
        description
          "The type of progress report provided.";
      }
      leaf message {
        type string;
        description
          "An optional arbitrary value.";
      }
      container ssh-host-keys {
        when "../progress-type = 'bootstrap-complete'" {
          description
            "SSH host keys are only sent when the progress type
             is 'bootstrap-complete'.";
        }
        description
          "A list of SSH host keys an NMS may use to authenticate
           subsequent SSH-based connections to this device (e.g.,
           netconf-ssh, netconf-ch-ssh).";
        list ssh-host-key {
          description
```

```
                 "An SSH host key an NMS may use to authenticate
                  subsequent SSH-based connections to this device
                  (e.g., netconf-ssh, netconf-ch-ssh).";
               reference
                 "RFC 4253: The Secure Shell (SSH) Transport Layer
                            Protocol";
               leaf algorithm {
                 type string;
                 mandatory true;
                 description
                   "The public key algorithm name for this SSH key.

                    Valid values are listed in the 'Public Key Algorithm
                    Names' subregistry of the 'Secure Shell (SSH) Protocol
                    Parameters' registry maintained by IANA.";
                 reference
                   "RFC 4250: The Secure Shell (SSH) Protocol Assigned
                              Numbers
                    IANA URL: https://www.iana.org/assignments/ssh-param\\
                              eters/ssh-parameters.xhtml#ssh-parameters-19
                              ('\\' added for formatting reasons)";
               }
               leaf key-data {
                 type binary;
                 mandatory true;
                 description
                   "The binary public key data for this SSH key, as
                    specified by RFC 4253, Section 6.6, i.e.:

                        string    certificate or public key format
                                  identifier
                        byte[n]   key/certificate data.";
                 reference
                   "RFC 4253: The Secure Shell (SSH) Transport Layer
                              Protocol";
               }
             }
           }
         }
         container trust-anchor-certs {
           when "../progress-type = 'bootstrap-complete'" {
             description
               "Trust anchors are only sent when the progress type
                is 'bootstrap-complete'.";
           }
           description
             "A list of trust anchor certificates an NMS may use to
              authenticate subsequent certificate-based connections
              to this device (e.g., restconf-tls, netconf-tls, or
```

```
                even netconf-ssh with X.509 support from RFC 6187).
                In practice, trust anchors for IDevID certificates do
                not need to be conveyed using this mechanism.";
            reference
              "RFC 6187:
                 X.509v3 Certificates for Secure Shell Authentication.";
            leaf-list trust-anchor-cert {
              type cms;
              description
               "A CMS structure whose top-most content type MUST be the
                signed-data content type, as described by Section 5 in
                RFC 5652.

                The CMS MUST contain the chain of X.509 certificates
                needed to authenticate the certificate presented by
                the device.

                The CMS MUST contain only a single chain of
                certificates.  The device's end-entity certificate
                MUST only authenticate to the last intermediate CA
                certificate listed in the chain.

                In all cases, the chain MUST include a self-signed
                root certificate.  In the case where the root
                certificate is itself the issuer of the device's
                end-entity certificate, only one certificate is
                present.

                This CMS encodes the degenerate form of the SignedData
                structure that is commonly used to disseminate X.509
                certificates and revocation objects (RFC 5280).";
              reference
               "RFC 5280:
                  Internet X.509 Public Key Infrastructure
                  Certificate and Certificate Revocation List (CRL)
                  Profile.
                RFC 5652:
                  Cryptographic Message Syntax (CMS)";
            }
          }
        }
      }
    }
    <CODE ENDS>
```

8. DHCP 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 (143) | option-length |
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
      .   .
      .   bootstrap-server-list (variable length)   .
      .   .
      +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

      o option-code: OPTION_V4_ZEROTOUCH_REDIRECT (143)
      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 Section 8.3.

   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 processes the response
   according to Section 5.5, with the understanding that the "address"
   and "port" values are encoded in the URIs.

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

As the list of URIs may exceed the maximum allowed length of a single
DHCPv4 option (255 octets), the client MUST implement [RFC3396],
allowing the URI list to be split across a number of
OPTION_V4_ZEROTOUCH_REDIRECT option instances.

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.

The server's DHCP message MUST contain only a single instance of the
OPTION_V4_ZEROTOUCH_REDIRECT's 'bootstrap-server-list' field.
However, the list of URIs in this field may exceed the maximum
allowed length of a single DHCPv4 option (per [RFC3396]).

If the length of 'bootstrap-server-list' is small enough to fit into
a single instance of OPTION_V4_ZEROTOUCH_REDIRECT, the server MUST
NOT send more than one instance of this option.

If the length of the 'bootstrap-server-list' field is too large to
fit into a single option, then OPTION_V4_ZEROTOUCH_REDIRECT MUST be
split into multiple instances of the option according to the process
described in [RFC3396].

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 (136)      |          option-length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.             bootstrap-server-list (variable length)          .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

o option-code: OPTION_V6_ZEROTOUCH_REDIRECT (136)
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 Section 8.3.

DHCPv6 Client Behavior

Clients MAY request the OPTION_V6_ZEROTOUCH_REDIRECT option, as
defined in [RFC8415], Sections 18.2.1, 18.2.2, 18.2.4, 18.2.5,
18.2.6, and 21.7.   As a convenience to the reader, we mention here
that the client includes requested option codes in the Option Request
Option.

On receipt of a DHCPv6 Reply message which contains the
OPTION_V6_ZEROTOUCH_REDIRECT, the client processes the response
according to Section 5.5, with the understanding that the "address"
and "port" values are encoded in the URIs.

Any invalid URI entries received in the uri-data field are ignored by
the client. If OPTION_V6_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

Section 18.3 of [RFC8415] governs server operation in regard to
option assignment. As a convenience to the reader, we mention here
that the server will send a particular option code only if configured
with specific values for that option code and if the client requested
it.

Option OPTION_V6_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). Each
URI entry in the bootstrap-server-list is structured as follows:

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

o uri-length: 2 octets long, specifies the length of the URI data.

o URI: URI of zerotouch bootstrap server, using the HTTPS URI
   scheme defined in Section 2.7.2 of RFC7230. URI MUST be in
   form "https://<ip-address-or-hostname>[:<port>]".

9. Security Considerations

9.1. Clock Sensitivity

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

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

Implementations SHOULD NOT rely on NTP for time, as NTP is not a
secure protocol at this time. Note, there is an IETF work-in-
progress to secure NTP [I-D.ietf-ntp-using-nts-for-ntp].

9.2. Use of IDevID Certificates

IDevID certificates, as defined in [Std-802.1AR-2009], are
RECOMMENDED, both for the TLS-level client certificate used by
devices when connecting to a bootstrap server, as well as for the
device identity certificate used by owners when encrypting the zero
touch artifacts.

9.3. Immutable Storage for Trust Anchors

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

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

9.4. Secure Storage for Long-lived Private Keys

Manufacturer-generated device identifiers may have very long
lifetimes. For instance, [Std-802.1AR-2009] recommends using the
"notAfter" value 99991231235959Z in IDevID certificates. Given the

long-lived nature of these private keys, it is paramount that they
are stored so as to resist discovery, such as in a secure
cryptographic processor, such as a trusted platform module (TPM)
chip.

9.5. Blindly Authenticating a Bootstrap Server

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

To compensate for this, this document requires that devices, when
connected to an untrusted bootstrap server, assert that data
downloaded from the server is signed.

9.6. Disclosing Information to Untrusted Servers

This document allows devices to establish connections to untrusted
bootstrap servers. However, since the bootstrap server is untrusted,
it may be under the control of an adversary, and therefore devices
SHOULD be cautious about the data they send to the bootstrap server
in such cases.

Devices send different data to bootstrap servers at each of the
protocol layers TCP, TLS, HTTP, and RESTCONF.

At the TCP protocol layer, devices may relay their IP address,
subject to network translations. Disclosure of this information is
not considered a security risk.

At the TLS protocol layer, devices may use a client certificate to
identify and authenticate themselves to untrusted bootstrap servers.
At a minimum, the client certificate must disclose the device's
serial number, and may disclose additional information such as the
device's manufacturer, hardware model, public key, etc. Knowledge of
this information may provide an adversary with details needed to
launch an attack. It is RECOMMENDED that secrecy of the network
constituency is not relied on for security.

At the HTTP protocol layer, devices may use an HTTP authentication
scheme to identify and authenticate themselves to untrusted bootstrap
servers. At a minimum, the authentication scheme must disclose the
device's serial number and, concerningly, may, depending on the
authentication mechanism used, reveal a secret that is only supposed
to be known to the device (e.g., a password). Devices SHOULD NOT use
an HTTP authentication scheme (e.g., HTTP Basic) with an untrusted

bootstrap server that reveals a secret that is only supposed to be
known to the device.

At the RESTCONF protocol layer, devices use the "get-bootstrapping-
data" RPC, but not the "report-progress" RPC, when connected to an
untrusted bootstrap server. The "get-bootstrapping-data" RPC allows
additional input parameters to be passed to the bootstrap server
(e.g., "os-name", "os-version", "hw-model"). It is RECOMMENDED that
devices only pass the "untrusted-connection" input parameter to an
untrusted bootstrap server. While it is okay for a bootstrap server
to immediately return signed onboarding information, it is
RECOMMENDED that bootstrap servers instead promote the untrusted
connection to a trusted connection, as described in Appendix B, thus
enabling the device to use the "report-progress" RPC while processing
the onboarding information.

9.7. Sequencing Sources of Bootstrapping Data

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

9.8. Safety of Private Keys used for Trust

The solution presented in this document enables bootstrapping data to
be trusted in two ways, either through transport level security or
through the signing of artifacts.

When transport level security (i.e., a trusted bootstrap server) is
used, the private key for the end-entity certificate must be online
in order to establish the TLS connection.

When artifacts are signed, the signing key is required to be online
only when the bootstrap server is returning a dynamically generated
signed-data response. For instance, a bootstrap server, upon
receiving the "untrusted-connection" input parameter to the "get-
bootstrapping-data" RPC, may dynamically generate a response that is
signed.

Bootstrap server administrators are RECOMMENDED to follow best
practice to protect the private key used for any online operation.
For instance, use of a hardware security module (HSM) is RECOMMENDED.
If an HSM is not used, frequent private key refreshes are
RECOMMENDED, assuming all bootstrapping devices have an accurate
clock (see Section 9.1).

For best security, it is RECOMMENDED that owners only provide
bootstrapping data that has been signed, using a protected private
key, and encrypted, using the device's public key from its secure
device identity certificate.

9.9. Increased Reliance on Manufacturers

The zero touch bootstrapping protocol presented in this document
shifts some control of initial configuration away from the rightful
owner of the device and towards the manufacturer and its delegates.

The manufacturer maintains the list of well-known bootstrap servers
its devices will trust. By design, if no bootstrapping data is found
via other methods first, the device will try to reach out to the
well-known bootstrap servers. There is no mechanism to prevent this
from occurring other than by using an external firewall to block such
connections. Concerns related to trusted bootstrap servers are
discussed in Section 9.10.

Similarly, the manufacturer maintains the list of voucher signing
authorities its devices will trust. The voucher signing authorities
issue the vouchers that enable a device to trust an owner's domain
certificate. It is vital that manufacturers ensure the integrity of
these voucher signing authorities, so as to avoid incorrect
assignments.

Operators should be aware that this system assumes that they trust
all the pre-configured bootstrap servers and voucher signing
authorities designated by the manufacturers. While operators may use
points in the network to block access to the well-known bootstrap
servers, operators cannot prevent voucher signing authorities from
generating vouchers for their devices.

9.10. Concerns with Trusted Bootstrap Servers

Trusted bootstrap servers, whether well-known or discovered, have the
potential to cause problems, such as the following.

o A trusted bootstrap server that has been compromised may be
   modified to return unsigned data of any sort. For instance, a
   bootstrap server that is only suppose to return redirect
   information might be modified to return onboarding information.
   Similarly, a bootstrap server that is only supposed to return
   signed data, may be modified to return unsigned data. In both
   cases, the device will accept the response, unaware that it wasn't
   supposed to be any different. It is RECOMMENDED that maintainers
   of trusted bootstrap servers ensure that their systems are not
   easily compromised and, in case of compromise, have mechanisms in

      place to detect and remediate the compromise as expediently as
      possible.

   o A trusted bootstrap server hosting either unsigned or signed but
      not encrypted data may disclose information to unwanted parties
      (e.g., an administrator of the bootstrap server). This is a
      privacy issue only, but could reveal information that might be
      used in a subsequent attack. Disclosure of redirect information
      has limited exposure (it is just a list of bootstrap servers),
      whereas disclosure of onboarding information could be highly
      revealing (e.g., network topology, firewall policies, etc.). It
      is RECOMMENDED that operators encrypt the bootstrapping data when
      its contents are considered sensitive, even to the administrators
      of a bootstrap server.

9.11. Validity Period for Zero Touch Information

   Zero touch information does not specify a validity period. For
   instance, neither redirect information nor onboarding information
   enable "not-before" or "not-after" values to be specified, and
   neither artifact alone can be revoked.

   For unsigned data provided by an untrusted source of bootstrapping
   data, it is not meaningful to discuss its validity period when the
   information itself has no authenticity and may have come from
   anywhere.

   For unsigned data provided by a trusted source of bootstrapping data
   (i.e., a bootstrap server), the availability of the data is the only
   measure of it being current. Since the untrusted data comes from a
   trusted source, its current availability is meaningful and, since
   bootstrap servers use TLS, the contents of the exchange cannot be
   modified or replayed.

   For signed data, whether provided by an untrusted or trusted source
   of bootstrapping data, the validity is constrained by the validity of
   the both the ownership voucher and owner certificate used to
   authenticate it.

   The ownership voucher's validity is primarily constrained by the
   ownership voucher's "created-on" and "expires-on" nodes. While
   [RFC8366] recommends short-lived vouchers (see Section 6.1), the
   "expires-on" node may be set to any point in the future, or omitted
   altogether to indicate that the voucher never expires. The ownership
   voucher's validity is secondarily constrained by the manufacturer's
   PKI used to sign the voucher; whilst an ownership voucher cannot be
   revoked directly, the PKI used to sign it may be.

The owner certificate's validity is primarily constrained by the
X.509's validity field, the "notBefore" and "notAfter" values, as
specified by the certificate authority that signed it. The owner
certificate's validity is secondarily constrained by the validity of
the PKI used to sign the voucher. Owner certificates may be revoked
directly.

For owners that wish to have maximum flexibility in their ability to
specify and constrain the validity of signed data, it is RECOMMENDED
that a unique owner certificate is created for each signed artifact.
Not only does this enable a validity period to be specified, for each
artifact, but it also enables to the validity of each artifact to be
revoked.

9.12.   The "ietf-zerotouch-information" YANG Module

The ietf-zerotouch-information module defined in this document
defines a data structure that is always wrapped by a CMS structure.
When accessed by a secure mechanism (e.g., protected by TLS), then
the CMS structure may be unsigned. However, when accessed by an
insecure mechanism (e.g., removable storage device), then the CMS
structure must be signed, in order for the device to trust it.

Implementations should be aware that signed bootstrapping data only
protects the data from modification, and that the contents are still
visible to others. This doesn't affect security so much as privacy.
That the contents may be read by unintended parties when accessed by
insecure mechanisms is considered next.

The ietf-zerotouch-information module defines a top-level "choice"
statement that declares the contents are either "redirect-
information" or "onboarding-information". Each of these two cases
are now considered.

When the content of the CMS structure is redirect-information, an
observer can learn about the bootstrap servers the device is being
directed to, their IP addresses or hostnames, ports, and trust anchor
certificates. Knowledge of this information could provide an
observer some insight into a network's inner structure.

When the content of the CMS structure is onboarding information, an
observer could learn considerable information about how the device is
to be provisioned. This information includes the operating system
version, initial configuration, and script contents. This
information should be considered sensitive and precautions should be
taken to protect it (e.g., encrypt the artifact using the device's
public key).

9.13. The "ietf-zerotouch-bootstrap-server" YANG Module

   The ietf-zerotouch-bootstrap-server module defined in this document
   specifies an API for a RESTCONF [RFC8040]. The lowest RESTCONF layer
   is HTTPS, and the mandatory-to-implement secure transport is TLS
   [RFC8446].

   The NETCONF Access Control Model (NACM) [RFC8341] provides the means
   to restrict access for particular users to a preconfigured subset of
   all available protocol operations and content.

   This module presents no data nodes (only RPCs). There is no need to
   discuss the sensitivity of data nodes.

   This module defines two RPC operations that may be considered
   sensitive in some network environments. These are the operations and
   their sensitivity/vulnerability:

   get-bootstrapping-data: This RPC is used by devices to obtain their
      bootstrapping data. By design, each device, as identified by its
      authentication credentials (e.g. client certificate), can only
      obtain its own data. NACM is not needed to further constrain
      access to this RPC.

   report-progress: This RPC is used by devices to report their
      bootstrapping progress. By design, each device, as identified by
      its authentication credentials (e.g. client certificate), can
      only report data for itself. NACM is not needed to further
      constrain access to this RPC.

10. IANA Considerations

10.1. The IETF XML Registry

   This document registers two URIs in the "ns" subregistry of the IETF
   XML Registry [RFC3688] maintained at
   https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns.
   Following the format in [RFC3688], the following registrations are
   requested:

   URI: urn:ietf:params:xml:ns:yang:ietf-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.

10.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names
registry [RFC6020] maintained at https://www.iana.org/assignments/
yang-parameters/yang-parameters.xhtml. Following the format defined
in [RFC6020], the below registrations are requested:

```
name:      ietf-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  (note: '\' used for formatting reasons only)
prefix:    ztbs
reference: RFC XXXX
```

10.3. The SMI Security for S/MIME CMS Content Type Registry

This document registers two SMI security codes in the "SMI Security
for S/MIME CMS Content Type" registry (1.2.840.113549.1.9.16.1)
maintained at https://www.iana.org/assignments/smi-numbers/smi-
numbers.xhtml#security-smime-1. Following the format used in
Section 3.4 of [RFC7107], the below registrations are requested:

```
Decimal  Description                             References
-------  --------------------------------------  ----------
TBD1     id-ct-zerotouchInformationXML           [RFCXXXX]
TBD2     id-ct-zerotouchInformationJSON          [RFCXXXX]
```

id-ct-zerotouchInformationXML indicates that the "zerotouch-
information" is encoded using XML. id-ct-zerotouchInformationJSON
indicates that the "zerotouch-information" is encoded using JSON.

10.4. The BOOTP Manufacturer Extensions and DHCP Options Registry

This document registers one DHCP code point in the "BOOTP
Manufacturer Extensions and DHCP Options" registry maintained at
http://www.iana.org/assignments/bootp-dhcp-parameters. Following the
format used by other registrations, the below registration is
requested:

```
Tag:          143
Name:         OPTION_V4_ZEROTOUCH_REDIRECT
Data Length: N
Meaning:      This option provides a list of URIs
              for zero touch bootstrap servers
Reference:   [RFCXXXX]
```

Note: this request is to make permanent a previously registered early code point allocation.

10.5. The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Registry

This document registers one DHCP code point in "Option Codes" subregistry of the "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" registry maintained at http://www.iana.org/assignments/dhcpv6-parameters. Following the format used by other registrations, the below registration is requested:

```
Value:            136
Description:      OPTION_V6_ZEROTOUCH_REDIRECT
Client ORO:       Yes
Singleton Option: Yes
Reference:        [RFCXXXX]
```

Note: this request is to make permanent a previously registered early code point allocation.

10.6. The Service Name and Transport Protocol Port Number Registry

This document registers one service name in the Service Name and Transport Protocol Port Number Registry [RFC6335] maintained at https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. Following the format defined in Section 8.1.1 of [RFC6335], the below registration is requested:

```
Service Name:            zerotouch
Transport Protocol(s):   TCP
Assignee:                IESG <iesg@ietf.org>
Contact:                 IETF Chair <chair@ietf.org>
Description:             This service name is used to construct the
                         SRV service label "_zerotouch" for discovery
                         of zero touch bootstrap servers.
Reference:               [RFCXXXX]
Port Number:             N/A
Service Code:            N/A
Known Unauthorized Uses: N/A
Assignment Notes:        This protocol uses HTTPS as a substrate.
```

11. References

11.1. Normative References

[I-D.ietf-netmod-yang-data-ext]
          Bierman, A., Bjorklund, M., and K. Watsen, "YANG Data
          Extensions", draft-ietf-netmod-yang-data-ext-01 (work in
          progress), March 2018.

[ITU.X690.2015]
          International Telecommunication Union, "Information
          Technology - ASN.1 encoding rules: Specification of Basic
          Encoding Rules (BER), Canonical Encoding Rules (CER) and
          Distinguished Encoding Rules (DER)", ITU-T Recommendation
          X.690, ISO/IEC 8825-1, August 2015,
          <https://www.itu.int/rec/T-REC-X.690/>.

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

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
          specifying the location of services (DNS SRV)", RFC 2782,
          DOI 10.17487/RFC2782, February 2000,
          <https://www.rfc-editor.org/info/rfc2782>.

[RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the
          Dynamic Host Configuration Protocol (DHCPv4)", RFC 3396,
          DOI 10.17487/RFC3396, November 2002,
          <https://www.rfc-editor.org/info/rfc3396>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH)
          Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253,
          January 2006, <https://www.rfc-editor.org/info/rfc4253>.

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

   [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70,
              RFC 5652, DOI 10.17487/RFC5652, September 2009,
              <https://www.rfc-editor.org/info/rfc5652>.

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

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

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

   [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>.

   [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
              Protocol (HTTP/1.1): Message Syntax and Routing",
              RFC 7230, DOI 10.17487/RFC7230, June 2014,
              <https://www.rfc-editor.org/info/rfc7230>.

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

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

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

   [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,
              "A Voucher Artifact for Bootstrapping Protocols",
              RFC 8366, DOI 10.17487/RFC8366, May 2018,
              <https://www.rfc-editor.org/info/rfc8366>.

   [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A.,
              Richardson, M., Jiang, S., Lemon, T., and T. Winters,
              "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)",
              RFC 8415, DOI 10.17487/RFC8415, November 2018,
              <https://www.rfc-editor.org/info/rfc8415>.

   [Std-802.1AR-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>.

11.2. Informative References

   [I-D.ietf-netconf-crypto-types]
              Watsen, K. and H. Wang, "Common YANG Data Types for
              Cryptography", draft-ietf-netconf-crypto-types-02 (work in
              progress), October 2018.

   [I-D.ietf-netconf-trust-anchors]
              Watsen, K., "YANG Data Model for Global Trust Anchors",
              draft-ietf-netconf-trust-anchors-02 (work in progress),
              October 2018.

   [I-D.ietf-ntp-using-nts-for-ntp]
              Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R.
              Sundblad, "Network Time Security for the Network Time
              Protocol", draft-ietf-ntp-using-nts-for-ntp-14 (work in
              progress), October 2018.

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

   [RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH)
              Protocol Assigned Numbers", RFC 4250,
              DOI 10.17487/RFC4250, January 2006,
              <https://www.rfc-editor.org/info/rfc4250>.

   [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure
              Shell Authentication", RFC 6187, DOI 10.17487/RFC6187,
              March 2011, <https://www.rfc-editor.org/info/rfc6187>.

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

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

   [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S.
              Cheshire, "Internet Assigned Numbers Authority (IANA)
              Procedures for the Management of the Service Name and
              Transport Protocol Port Number Registry", BCP 165,
              RFC 6335, DOI 10.17487/RFC6335, August 2011,
              <https://www.rfc-editor.org/info/rfc6335>.

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

   [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms
              for DNS (EDNS(0))", STD 75, RFC 6891,
              DOI 10.17487/RFC6891, April 2013,
              <https://www.rfc-editor.org/info/rfc6891>.

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

   [RFC7107] Housley, R., "Object Identifier Registry for the S/MIME
              Mail Security Working Group", RFC 7107,
              DOI 10.17487/RFC7107, January 2014,
              <https://www.rfc-editor.org/info/rfc7107>.

   [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and
              D. Wessels, "DNS Transport over TCP - Implementation
              Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016,
              <https://www.rfc-editor.org/info/rfc7766>.

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

   [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
              BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
              <https://www.rfc-editor.org/info/rfc8340>.

   [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration
              Access Control Model", STD 91, RFC 8341,
              DOI 10.17487/RFC8341, March 2018,
              <https://www.rfc-editor.org/info/rfc8341>.

   [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
              <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. The Zero Touch Device Data Model

   This section defines a non-normative data model that enables the
   configuration of zerotouch bootstrapping and discovery of what
   parameters are used by a device's bootstrapping logic.

A.1. Data Model Overview

   The following tree diagram provides an overview for the zerotouch
   device data model.

```
module: example-zerotouch-device
  +--rw zerotouch
     +--rw enabled?                            boolean
     +--ro idevid-certificate?
     |       ct:end-entity-cert-cms {bootstrap-servers}?
     +--ro bootstrap-servers {bootstrap-servers}?
     |  +--ro bootstrap-server* [address]
     |     +--ro address    inet:host
     |     +--ro port?      inet:port-number
     +--ro bootstrap-server-pinned-certificates?
     |       ta:pinned-certificates-ref {bootstrap-servers}?
     +--ro voucher-pinned-certificates?
             ta:pinned-certificates-ref {signed-data}?
```

   In the above diagram, notice that there is only one configurable node
   "enabled". The expectation is that this node would be set to "true"
   in device's factory default configuration and that it would either be
   set to "false" or deleted when the zerotouch bootstrapping is longer
   needed.

A.2. Example Usage

   Following is an instance example for this data model.

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

   <zerotouch
     xmlns="https://example.com/zerotouch-device">
     <enabled>true</enabled>
     <idevid-certificate>base64encodedvalue==</idevid-certificate>
     <bootstrap-servers>
       <bootstrap-server>
         <address>phs1.example.com</address>
         <port>8443</port>
       </bootstrap-server>
       <bootstrap-server>
         <address>phs2.example.com</address>
         <port>8443</port>
       </bootstrap-server>
       <bootstrap-server>
         <address>phs3.example.com</address>
         <port>8443</port>
       </bootstrap-server>
     </bootstrap-servers>
     <bootstrap-server-pinned-certificates>manufacturers-root-ca-certs<\
   /bootstrap-server-pinned-certificates>
     <voucher-pinned-certificates>manufacturers-root-ca-certs</voucher-\
   pinned-certificates>
   </zerotouch>
```

A.3. YANG Module

The device model is defined by the YANG module defined in this
section.

This module uses data types defined in [RFC6991],
[I-D.ietf-netconf-crypto-types], and
[I-D.ietf-netconf-trust-anchors].

```
   module example-zerotouch-device {
     yang-version 1.1;
     namespace "https://example.com/zerotouch-device";
     prefix ztd;

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

     import ietf-crypto-types {
       prefix ct;
       revision-date 2018-06-04;
```

```
          description
           "This revision is defined in the -00 version of
            draft-ietf-netconf-crypto-types";
          reference
           "draft-ietf-netconf-crypto-types:
              Common YANG Data Types for Cryptography";
        }

        import ietf-trust-anchors {
          prefix ta;
          revision-date 2018-06-04;
          description
           "This revision is defined in -00 version of
            draft-ietf-netconf-trust-anchors.";
          reference
           "draft-ietf-netconf-trust-anchors:
              YANG Data Model for Global Trust Anchors";
        }

        organization
          "Example Corporation";

        contact
          "Author: Bootstrap Admin <mailto:admin@example.com>";

        description
          "This module defines a data model to enable zerotouch
           bootstrapping and discover what parameters are used.
           This module assumes the use of an IDevID certificate,
           as opposed to any other client certificate, or the
           use of an HTTP-based client authentication scheme.";

        revision 2018-12-20 {
          description
            "Initial version";
          reference
            "RFC XXXX: Zero Touch Provisioning for Networking Devices";
        }

        // features

        feature bootstrap-servers {
          description
            "The device supports bootstrapping off bootstrap servers.";
        }

        feature signed-data {
          description
```

```
        "The device supports bootstrapping off signed data.";
      }

      // protocol accessible nodes

      container zerotouch {
        description
          "Top-level container for zerotouch data model.";
        leaf enabled {
          type boolean;
          default false;
          description
            "The 'enabled' leaf controls if zerotouch bootstrapping is
             enabled or disabled.  The default is 'false' so that, when
             not enabled, which is most of the time, no configuration
             is needed.";
        }
        leaf idevid-certificate {
          if-feature bootstrap-servers;
          type ct:end-entity-cert-cms;
          config false;
          description
            "This CMS structure contains the IEEE 802.1AR-2009
             IDevID certificate itself, and all intermediate
             certificates leading up to, and optionally including,
             the manufacturer's well-known trust anchor certificate
             for IDevID certificates.  The well-known trust anchor
             does not have to be a self-signed certificate.";
          reference
            "IEEE 802.1AR-2009:
                IEEE Standard for Local and metropolitan area
                networks - Secure Device Identity.";
        }
        container bootstrap-servers {
          if-feature bootstrap-servers;
          config false;
          description
            "List of bootstrap servers this device will attempt
             to reach out to when bootstrapping.";
          list bootstrap-server {
            key "address";
            description
              "A bootstrap server entry.";
            leaf address {
              type inet:host;
              mandatory true;
              description
                  "The IP address or hostname of the bootstrap server the
```

```
                    device should redirect to.";
              }
              leaf port {
                type inet:port-number;
                default "443";
                description
                  "The port number the bootstrap server listens on.  If no
                   port is specified, the IANA-assigned port for 'https'
                   (443) is used.";
              }
            }
          }
          leaf bootstrap-server-pinned-certificates {
            if-feature bootstrap-servers;
            type ta:pinned-certificates-ref;
            config false;
            description
              "A reference to a list of pinned certificate authority (CA)
               certificates that the device uses to validate bootstrap
               servers with.";
          }
          leaf voucher-pinned-certificates {
            if-feature signed-data;
            type ta:pinned-certificates-ref;
            config false;
            description
              "A reference to a list of pinned certificate authority (CA)
               certificates that the device uses to validate ownership
               vouchers with.";
          }
        }
      }
    }
```

Appendix B. Promoting a Connection from Untrusted to Trusted

   The following diagram illustrates a sequence of bootstrapping
   activities that promote an untrusted connection to a bootstrap server
   to a trusted connection to the same bootstrap server. This enables a
   device to limit the amount of information it might disclose to an
   adversary hosting an untrusted bootstrap server.

```
                                            +----------+
                                            |Deployment|
                                            | Specific |
       +------+                             |Bootstrap |
       |Device|                             |  Server  |
       +------+                             +----------+
          |                                     |
          | 1. "HTTPS" Request ("untrusted-connection", nonce)    |
          |---------------------------------------------------->|
          | 2. "HTTPS" Response (signed redirect information)    |
          |<----------------------------------------------------|
          |                                     |
          |                                     |
          | 3. HTTPS Request (os-name=xyz, os-version=123, etc.) |
          |---------------------------------------------------->|
          | 4. HTTPS Response (unsigned onboarding information   |
          |<----------------------------------------------------|
          |                                     |
```

The interactions in the above diagram are described below.

1. The device initiates an untrusted connection to a bootstrap
    server, as is indicated by putting "HTTPS" in double quotes
    above. It is still an HTTPS connection, but the device is unable
    to authenticate the bootstrap server's TLS certificate. Because
    the device is unable to trust the bootstrap server, it sends the
    "untrusted-connection" input parameter, and optionally also the
    "nonce" input parameter, in the "get-bootstrapping-data" RPC.
    The "untrusted-connection" parameter informs the bootstrap server
    that the device does not trust it and may be holding back some
    additional input parameters from the server (e.g., other input
    parameters, progress reports, etc.). The "nonce" input parameter
    enables the bootstrap server to dynamically obtain an ownership
    voucher from a MASA, which may be important for devices that do
    not have a reliable clock.

2. The bootstrap server, seeing the "untrusted-connection" input
    parameter, knows that it can either send unsigned redirect
    information or signed data of any type. But, in this case, the
    bootstrap server has the ability to sign data and chooses to
    respond with signed redirect information, not signed onboarding
    information as might be expected, securely redirecting the device
    back to it again. Not displayed but, if the "nonce" input
    parameter was passed, the bootstrap server could dynamically
    connect to a download a voucher from the MASA having the nonce
    value in it. Details regarding a protocol enabling this
    integration is outside the scope of this document.

3. Upon validating the signed redirect information, the device
    establishes a secure connection to the bootstrap server.
    Unbeknownst to the device, it is the same bootstrap server it was
    connected to previously but, because the device is able to
    authenticate the bootstrap server this time, it sends its normal
    "get-bootstrapping-data" request (i.e., with additional input
    parameters) as well as its progress reports (not depicted).

4. This time, because the "untrusted-connection" parameter was not
    passed, having access to all of the device's input parameters,
    the bootstrap server returns, in this example, unsigned
    onboarding information to the device. Note also that, because
    the bootstrap server is now trusted, the device will send
    progress reports to the server.

Appendix C. Workflow Overview

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

C.1. Enrollment and Ordering Devices

The following diagram illustrates key interactions that may occur
from when a prospective owner enrolls in a manufacturer's 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 |
          |                           #------------------------>|
          |                        |                         |
```
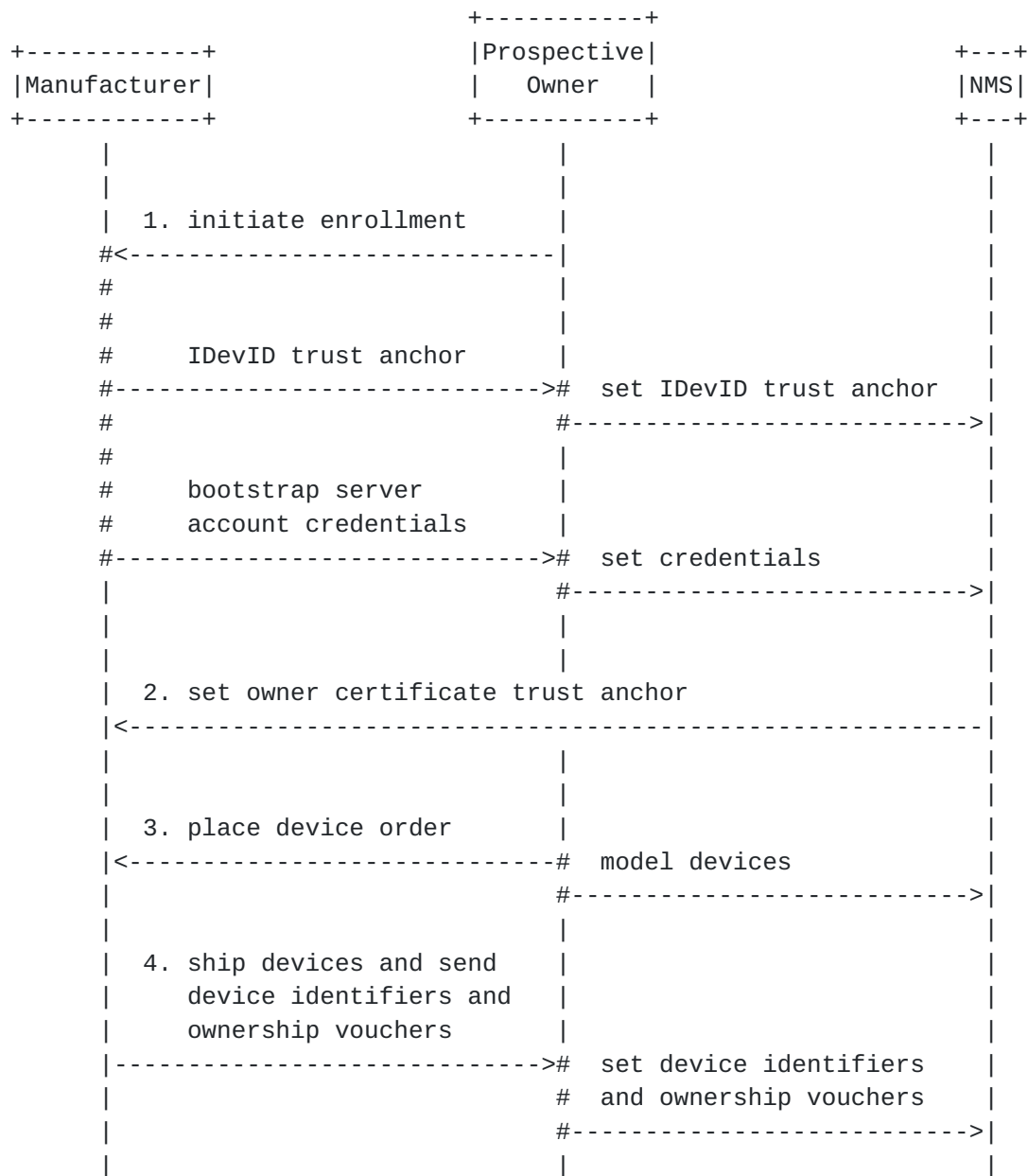
   Each numbered item below corresponds to a numbered item in the
   diagram above.

   1. A prospective owner of a manufacturer's devices initiates an
       enrollment process with the manufacturer. This process includes
       the following:

       * Regardless how the prospective owner intends to bootstrap
          their devices, they will always obtain from the manufacturer
          the trust anchor certificate for the IDevID certificates.
          This certificate will is installed on the prospective owner's

NMS so that the NMS can authenticate the IDevID certificates
when they are presented to subsequent steps.

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

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

3. Some time later, the prospective owner places an order with the
    manufacturer, perhaps with a special flag checked for 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 fulfills the order, shipping the devices to
    their intended locations, they may notify the owner of the
    devices' serial numbers and shipping destinations, which the
    owner may use to stage the network for when the devices power on.
    Additionally, the manufacturer may send one or more ownership
    vouchers, cryptographically assigning ownership of those devices
    to the owner. The owner may set this information on their NMS,
    perhaps binding specific modeled devices to the serial numbers
    and ownership vouchers.

C.2. Owner Stages the Network for Bootstrap

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

```
            +----------+ +------------+
            |Deployment| |Manufacturer| +------+ +------+
            | Specific | |   Hosted   | | Local| | Local| +----------+
     +---+ |Bootstrap | | Bootstrap  | |  DNS | | DHCP | |Removable|
     |NMS| |  Server  | |   Server   | |Server| |Server| | Storage |
     +---+ +----------+ +------------+ +------+ +------+ +----------+
       |        |             |            |        |         |
 1.    |        |             |            |        |         |
 activate|      |             |            |        |         |
 modeled |      |             |            |        |         |
 device  |      |             |            |        |         |
 ------->|      |             |            |        |         |
         | 2. (optional)      |            |        |         |
         |    configure       |            |        |         |
         |    bootstrap       |            |        |         |
         |    server          |            |        |         |
         |------->|           |            |        |         |
         |        |           |            |        |         |
         | 3. (optional) configure         |        |         |
         |    bootstrap server |           |        |         |
         |-------------------->|           |        |         |
         |        |            |           |        |         |
         |        |            |           |        |         |
         | 4. (optional) configure DNS server|       |         |
         |---------------------------------->|       |         |
         |        |            |           |        |         |
         |        |            |           |        |         |
         | 5. (optional) configure DHCP server        |         |
         |--------------------------------------------->|         |
         |        |            |           |        |         |
         |        |            |           |        |         |
         | 6. (optional) store bootstrapping artifacts on media |
         |----------------------------------------------------->|
         |        |            |           |        |         |
         |        |            |           |        |         |
```

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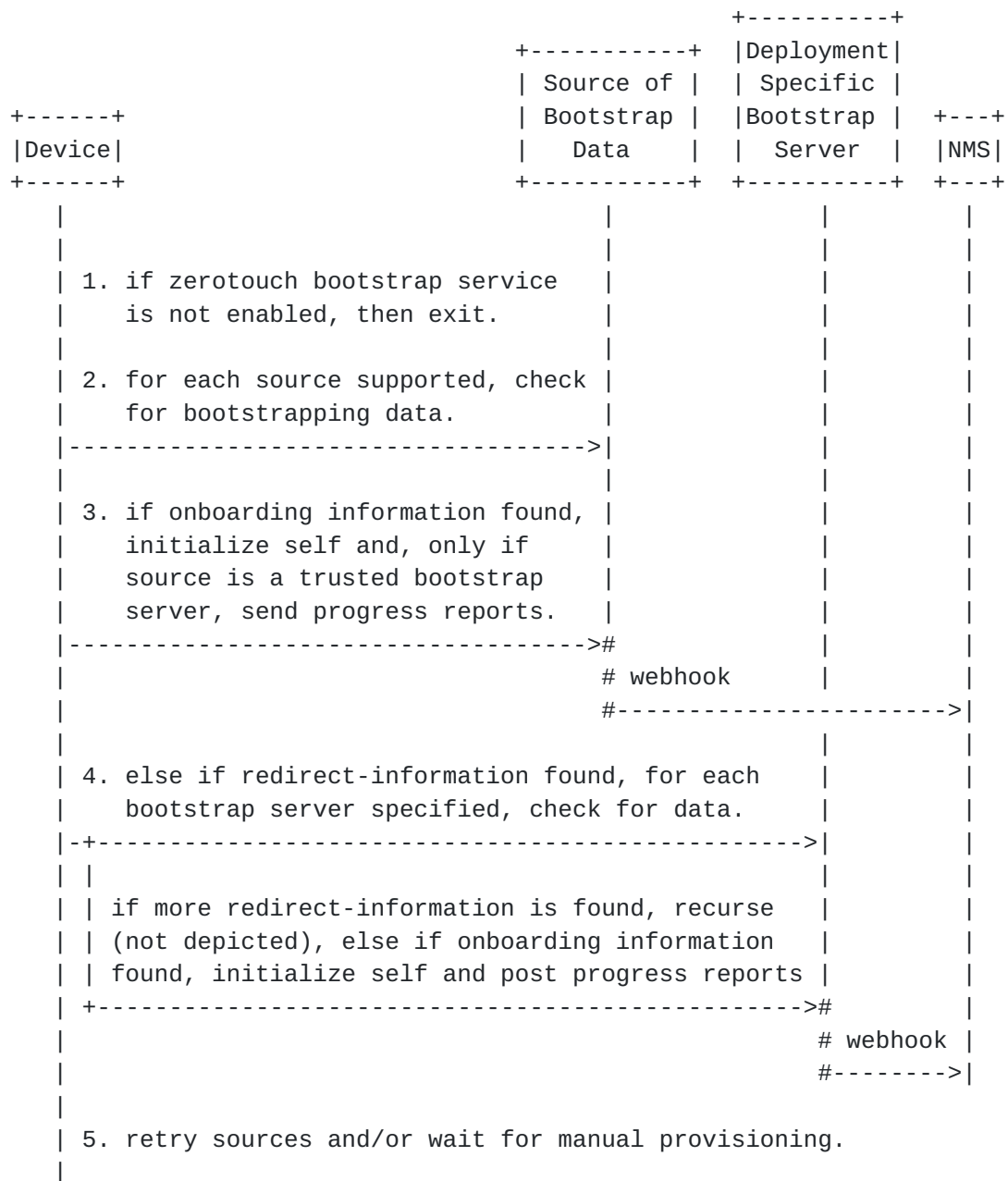   diagram above.

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

is depicted that an NMS performs the steps, but they may be
performed manually or through some other mechanism.

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

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

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

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

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

C.3. Device Powers On

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

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

   The interactions in the above diagram are described below.

   1. Upon power being applied, the device checks to see if zerotouch
       bootstrapping is configured, such as must be 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 reports to the
       bootstrap server.

       * The contents of the initial configuration should configure an
          administrator account on the device (e.g., username, SSH
          public key, etc.), and should configure the device either to
          listen for NETCONF or RESTCONF connections or to initiate call
          home connections [RFC8071], and should disable the zerotouch
          bootstrapping service (e.g., the "enabled" leaf in data model
          presented in Appendix A).

       * If the bootstrap server supports forwarding device progress
          reports to external systems (e.g., via a webhook), a
          "bootstrap-complete" progress report (Section 7.3) informs the
          external system to know when it can, for instance, initiate a
          connection to the device. To support this scenario further,
          the "bootstrap-complete" progress report may also relay the
          device's SSH host keys and/or TLS certificates, with which the
          external system can use to authenticate subsequent connections
          to the device.

       If the device successfully completes the bootstrapping process,
       it exits the bootstrapping logic without considering any
       additional sources of bootstrapping data.

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

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

Appendix D. Change Log

D.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

D.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

D.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

D.4. **02 to 03**

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

D.5. **03 to 04**

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

o Added many terms to Terminology section.

o Added all new "Guiding Principles" section.

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

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

D.6. **04 to 05**

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

o Created new section for information types

o Added support for DNS servers

o Now allows provisional TLS connections

o Bootstrapping data now supports scripts

o Device Details section overhauled

o Security Considerations expanded

o Filled in enumerations for notification types

D.7. **05 to 06**

   o Minor update

   o Added many Normative and Informative references.

   o Added new section Other Considerations.

D.8. **06 to 07**

   o Minor update

   o Added an Editorial Note section for RFC Editor.

   o Updated the IANA Considerations section.

D.9. **07 to 08**

   o Minor update

   o Updated to reflect review from Michael Richardson.

D.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).

D.11. **09 to 10**

   o Factored ownership voucher 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.

D.12. 10 to 11

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

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

D.13. 11 to 12

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

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

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

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

D.14. 12 to 13

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

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

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

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

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

D.15. 13 to 14

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

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

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

o Removed the Guiding Principles section.

D.16. 14 to 15

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

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

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

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

o Renamed "bootstrap information" to "update information".

o Removed "Other Considerations" section.

o Tons of editorial updates.

D.17. 15 to 16

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

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

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

o   added a note that, in order to simplify the verification process,
    the "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 .

D.18. 16 to 17

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

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

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

D.19. 17 to 18

o   Added Security Considerations for each YANG module.

o   Reverted back to the device always sending its DevID cert.

o   Moved data tree to "get-bootstrapping-data" RPC.

o   Moved the "update-progress" action to a "report-progress" RPC.

   o Added an "untrusted-connection" parameter to "get-bootstrapping-
      data" RPC.

   o Added the "ietf-zerotouch-device" module.

   o Lots of small updates.

D.20. 18 to 19

   o Fixed "must" expressions, by converting "choice" to a "list" of
      "image-verification", each of which now points to a base identity
      called "hash-algorithm". There's just one algorithm currently
      defined (sha-256). Wish there was a standard crypto module that
      could identify such identities.

D.21. 19 to 20

   o Now references I-D.ietf-netmod-yang-tree-diagrams.

   o Fixed tree-diagrams in Section 2 to always reflect current YANG
      (now they are now dynamically generated).

   o The "redirect-information" container's "trust-anchor" is now a CMS
      structure that can contain a chain of certificates, rather than a
      single certificate.

   o The "onboarding-information" container's support for image
      verification reworked to be extensible.

   o Added a reference to the "Device Details" section to the new
      example-zerotouch-device module.

   o Clarified that the device must always pass its IDevID certificate,
      even for untrusted bootstrap servers.

   o Fixed the description statement for the "script" typedef to refer
      to the [pre/post]-script-[warning/error] enums, rather than the
      legacy script-[warning/error] enums.

   o For the get-bootstrapping-data RPC's input, removed the "remote-
      id" and "circuit-id" fields, and added a "hw-model" field.

   o Improved DHCP error handling text.

   o Added MUST requirement for DHCPv6 client and server implementing
      [RFC3396] to handle URI lists longer than 255 octets.

   o Changed the "configuration" value in onboarding-information to be
      type "binary" instead of "anydata".

   o Moved everything from PKCS#7 to CMS (this shows up as a big
      change).

   o Added the early code point allocation assignments for the DHCP
      Options in the IANA Considerations section, and updated the RFC
      Editor note accordingly.

   o Added RFC Editor request to replace the assigned values for the
      CMS content types.

   o Relaxed auth requirements from device needing to always send
      IDevID cert to device needing to always send authentication
      credentials, as this better matches what RFC 8040 Section 2.5
      says.

   o Moved normative module "ietf-zerotouch-device" to non-normative
      module "example-zerotouch-device".

   o Updated Title, Abstract, and Introduction per discussion on list.

D.22. 20 to 21

   o Now any of the three artifact can be encrypted.

   o Fixed some line-too-long issues.

D.23. 21 to 22

   o Removed specifics around how scripts indicate warnings or errors
      and how scripts emit output.

   o Moved the Zero Touch Device Data Model section to the Appendix.

   o Modified the YANG module in the Zero Touch Device Data Model
      section to reflect the latest trust-anchors and keystore drafts.

   o Modified types in other YANG modules to more closely emulate what
      is in draft-ietf-netconf-crypto-types.

D.24. 22 to 23

   o Rewrote section 5.6 (processing onboboarding information) to be
      clearer about error handling and retained state. Specifically:

* Clarified that a script, upon having an error, must gracefully exit, cleaning up any state that might hinder subsequent executions.

* Added ability for scripts to be executed again with a flag enabling them to clean up state from a previous execution.

* Clarified that the conifguration commit is atomic.

* Clarified that any error encountered after committing the configuration (e.g., in the "post-configuration-script") must rollback the configuration to the previous configuration.

* Clarified that failure to successfully deliver the "bootstrap-initiated" and "bootstrap-complete" progress types must be treated as an error.

* Clarified that "return to bootstrapping sequence" is to be interpreted in the recursive context. Meaning that the device rolls-back one loop, rather than start over from scratch.

o Changed how a device verifies a boot-image from just "MUST match one of the supplied fingerprints" to also allow for the verification to use an cryptographic signature embedded into the image itself.

o Added more "progress-type" enums for visibility reasons, enabling more strongly-typed debug information to be sent to the bootstrap server.

o Added Security Considerations based on early SecDir review.

o Added recommendation for device to send warning if the initial config does not disable the bootstrapping process.

D.25. **23 to 24**

o Follow-ups from SecDir and Shepherd.

o Added "boot-image-complete" enumeration.

D.26. **24 to 25**

o Removed remaining old "bootstrapping information" term usage.

o Fixed DHCP Option length definition.

o Added reference to RFC 6187.

D.27. 25 to 26

o Updated URI structure text (sec 8.3) and added norm. ref to
   RFC7230 reflecting Alexey Melnikov's comment.

o Added IANA registration for the 'zerotouch' service, per IESG
   review from Adam Roach.

o Clarified device's looping behavior and support for alternative
   provisioning mechanisms, per IESG review from Mirja Kuehlewind.

o Updated "ietf-zerotouch-bootstrap-server:ssh-host-key" from leaf-
   list to list, per IESG review from Benjamin Kaduk.

o Added option size text to DHCPv4 option size to address Suresh
   Krishnan's IESG review discuss point.

o Updated RFC3315 to RFC8415 and associated section references.

o Revamped the DNS Server section, after digging into Alexey
   Melnikov comment.

o Fixed IETF terminology template section in both YANG modules.

Acknowledgements

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