                 Secure Zero Touch Provisioning (SZTP)

Abstract

   This document presents a technique to securely provision a networking
   device when it is booting in a factory-default state.  Variations in
   the solution enable 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.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF).  It represents the consensus of the IETF community.  It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG).  Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8572.

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
   device may establish NETCONF [RFC6241] and/or RESTCONF [RFC8040]
   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 alphabetically):

Artifact:  The term "artifact" is used throughout this document to
represent any of the three artifacts defined in Section 3
(conveyed information, ownership voucher, and owner certificate).
These artifacts collectively provide all the bootstrapping data a
device may use.

Bootstrapping Data:  The term "bootstrapping data" is used throughout
this document to refer to the collection of data that a device
may obtain during the bootstrapping process.  Specifically, it
refers to the three artifacts defined in Section 3 (conveyed
information, owner certificate, and ownership voucher).

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.

Conveyed Information:  The term "conveyed information" is used herein
to refer to either redirect information or onboarding
information.  Conveyed information is one of the three
bootstrapping artifacts described in Section 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 "conveyed information"
      defined in this document, the other being "redirect information".
      Onboarding information is formally defined by the "onboarding-
      information" container within the "conveyed-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 conveyed 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 "conveyed information"
      defined in this document, the other being "onboarding
      information".  Redirect information is formally defined by the
      "redirect-information" container within the "conveyed-
      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
      conveyed 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
      conveyed information that has not been signed.

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 Conveyed Information

   This document defines two types of conveyed information that devices
   can access during the bootstrapping process.  These conveyed
   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, an initial
   configuration the device must commit, and 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.  Conveyed Information

   The conveyed 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 conveyed information artifact is a Cryptographic Message Syntax
   (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 conveyed 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"; see Section 1.2.

When the conveyed information artifact is unsigned and unencrypted,
as it might be when communicated over trusted channels, the CMS
structure's topmost content type MUST be one of the OIDs described in
Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or
id-ct-sztpConveyedInfoJSON) 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
"conveyed-information" data in the expected encoding.

When the conveyed information artifact is unsigned and encrypted, as
it might be when communicated over trusted channels but, for some
reason, the operator wants to ensure that only the device is able to
see the contents, the CMS structure's topmost content type MUST be
the OID id-envelopedData (1.2.840.113549.1.7.3).  Furthermore, the
encryptedContentInfo's content type MUST be one of the OIDs described
in Section 10.3 (i.e., id-ct-sztpConveyedInfoXML or
id-ct-sztpConveyedInfoJSON) 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
"conveyed-information" data in the expected encoding.

When the conveyed information artifact is signed and unencrypted, as
it might be when communicated over untrusted channels, the CMS
structure's topmost 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-sztpConveyedInfoXML or id-ct-sztpConveyedInfoJSON) 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 "conveyed-information" data in the expected encoding.

When the conveyed information artifact is signed and encrypted, as it
might be when communicated over untrusted channels and privacy is
important, the CMS structure's topmost 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-sztpConveyedInfoXML or
id-ct-sztpConveyedInfoJSON), 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
   "conveyed-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 MUST have no Key Usage specified, or the Key Usage
   MUST, at a minimum, 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 conveyed 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 conveyed
   information artifact.

   The owner certificate artifact is formally a CMS structure, as
   specified by [RFC5652], encoded using ASN.1 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., Certificate
   Revocation Lists (CRLs) [RFC5280] and OCSP Responses [RFC6960]).
   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 Online Certificate Status
   Protocol (OCSP) responder specified in the associated certificates.

   When unencrypted, the topmost content type of the owner certificate
   artifact's CMS structure 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 topmost content type of the owner certificate
   artifact's CMS structure 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 topmost 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 topmost content type of the ownership voucher
artifact's CMS structure 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 conveyed information and ownership voucher
   artifacts are signed and then encrypted; they are never encrypted and
   then signed.

   This sequencing has the following advantages: 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 conveyed 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 revocations either are not needed or 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 when revocations are needed but 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"
   and "no" indicate whether or not the artifact is present in the
   artifact grouping):

| Artifact Grouping | Conveyed 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 SZTP 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:

      Conveyed Information:  Mapped to a file containing the binary
         artifact described in Section 3.1 (e.g., conveyed-
         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 that
   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 SZTP 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 third 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, e.g., from 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 and then, only if
   doing so does not result in a successful bootstrap, MUST query for
   device-independent DNS records.

   For each of the device-specific and device-independent queries,
   devices MUST first query using multicast DNS [RFC6762] and then, only
   if doing so does not result in a successful bootstrap, MUST query
   again using unicast DNS [RFC1035] [RFC7766].  This assumes the
   address of a DNS server is known, such as it may be using techniques
   similar to those described in Section 11 of [RFC6763].

   When querying for device-specific DNS records, devices MUST query for
   TXT records [RFC1035] under "<serial-number>._sztp", where <serial-
   number> is the device's serial number (the same value as in the
   device's secure device identity certificate), and "_sztp" is the
   globally scoped DNS attribute registered per this document (see
   Section 10.7).

   Example device-specific DNS record queries:

      TXT in <serial-number>._sztp.local.  (multicast)
      TXT in <serial-number>._sztp.<domain>.  (unicast)

   When querying for device-independent DNS records, devices MUST query
   for SRV records [RFC2782] under "_sztp._tcp", where "_sztp" is the
   service name registered per this document (see Section 10.6), and
   "_tcp" is the globally scoped DNS attribute registered per [RFC8552].

   Note that a device-independent response is only able to encode
   unsigned data anyway, since signed data necessitates the use of a
   device-specific ownership voucher.  Use of SRV records maximumly
   leverages existing DNS standards.  A response containing multiple SRV
   records is comparable to an unsigned redirect information's list of
   bootstrap servers.

   Example device-independent DNS record queries:

      SRV in _sztp._tcp.local.  (multicast)
      SRV in _sztp._tcp.<domain>.  (unicast)

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, similar to the technique described in Section 6.3 of
   [RFC6763].

   Artifact to TXT Record Mapping:

      Conveyed Information:  Mapped to a TXT record having the key "ci"
         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 "<serial-number>", the
   domain is "example.com", and "<binary data>" represents the binary
   artifact:

     <serial-number>._sztp.example.com. 3600 IN TXT "ci=<binary data>"
     <serial-number>._sztp.example.com. 3600 IN TXT "oc=<binary data>"
     <serial-number>._sztp.example.com. 3600 IN TXT "ov=<binary data>"

   Note that, in the case that "ci" 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:

      Conveyed 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
         never encode signed data; hence, there is no need for an owner
         certificate artifact.

      Ownership Voucher:  Not supported.  Device-independent responses
         never encode signed data; 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.  This example assumes
   that the domain is "example.com" and that there are four bootstrap
   servers "sztp[1-4]":

      _sztp._tcp.example.com. 1800 IN SRV 0 0 443 sztp1.example.com.
      _sztp._tcp.example.com. 1800 IN SRV 1 0 443 sztp2.example.com.
      _sztp._tcp.example.com. 1800 IN SRV 2 0 443 sztp3.example.com.
      _sztp._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.

   While this document does not use DNS-SD [RFC6763], per Section 12.2
   of that RFC, Multicast DNS (mDNS) responses SHOULD also include all
   address records (type "A" and "AAAA") named in the SRV rdata.

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 of
   [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
   Extension Mechanisms for DNS (EDNS(0)) extensions [RFC6891] enabled.
   Depending on content, signed data may also not fit into a multicast
   DNS packet, which bounds the size to 9000 bytes, per Section 17 of

    [RFC6762].  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 SZTP 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 third party.

    A DHCP server is an untrusted source of bootstrapping data.  Thus,
    the information stored on the DHCP server either MUST be signed or
    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:

       Conveyed 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 SZTP 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 provided in 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 Secure Shell (SSH)
   host keys and/or TLS trust anchor certificates).

   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 conveyed information returned from
   it MAY be signed.  When the bootstrap server is untrusted, the
   conveyed 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.

   Artifact to Bootstrap Server Mapping:

      Conveyed Information:  Mapped to the "conveyed-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.

   SZTP 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 pre-configured state and bootstrapping logic
   described in the following sections.

5.1.  Initial State

```
        +---------------------------------------------------------------+
        |                          <device>                             |
        |                                                               |
        |   +-------------------------------------------------------+   |
        |   |                   <read/write storage>                |   |
        |   |                                                       |   |
        |   | 1.  flag to enable SZTP 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 SZTP 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 SZTP bootstrapping.  This variable MUST be enabled by
       default in order for SZTP bootstrapping to run when the device
       first powers on.  Because it is a goal that the configuration
       installed by the bootstrapping process disables SZTP
       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 of
       [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 SZTP 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.  SZTP 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
```
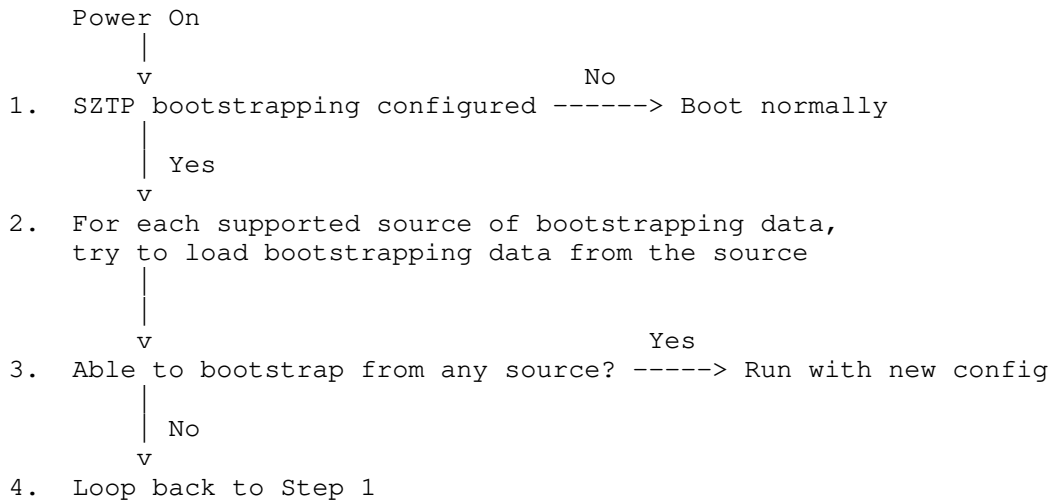
Note: At any time, the device MAY be configured via an alternate
      provisioning mechanism (e.g., command-line interface (CLI)).

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 SZTP
    bootstrapping is configured, as is expected to be the case for
    the device's pre-configured initial state.  If SZTP 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 process 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 CLI, a web-based user interface, or even another bootstrapping
   protocol.  Regardless of how it is configured, the configuration
   SHOULD unset the flag enabling SZTP bootstrapping as 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 globally 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 pre-configured 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 of [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 conveyed 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 that this is worded to cover the
special case when signed data is returned even from a trusted source
of bootstrapping data.

If the conveyed 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 conveyed 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.  Whether the processing of the onboarding

information succeeds or fails, the device MUST exit the recursive
algorithm, returning to the bootstrapping sequence described in
Section 5.2; the only difference is how it responds to the "Able to
bootstrap from any source?" conditional described in the figure in
that 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 pre-
   configured 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 a
   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 that the conveyed 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 a terminating "boot-image-installed-rebooting",
"bootstrap-complete", or error-specific progress report.  If the
"reporting-level" node of the bootstrap server's "get-bootstrapping-
data" RPC-reply is the value "verbose", the device MUST additionally
send all appropriate non-terminating progress reports (e.g.,
initiated, warning, complete, etc.).  Regardless of the reporting
level requested by the bootstrap server, the device MAY send progress
reports beyond those required by the 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, even though the onboarding information was,
necessarily, signed and authenticated.  Please be aware that
bootstrap servers are recommended to promote untrusted connections to
trusted connections, in the last paragraph of Section 9.6, so as to,
in part, be able to collect progress reports from devices.

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 use either
one of the verification fingerprints supplied by the onboarding
information or 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 SZTP 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.

      If the implementation chooses to undo previous steps, the
      following guidelines apply:

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

      *  Most steps are atomic.  For example, the processing of a
         configuration is atomic (as specified above), and the
         processing of scripts is atomic (as specified in the "ietf-
         sztp-conveyed-info" 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 Conveyed Information Data Model

   This section defines a YANG 1.1 [RFC7950] module that is used to
   define the data model for the conveyed information artifact described
   in Section 3.1.  This data model uses the "yang-data" extension
   statement defined in [RFC8040].  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 conveyed information artifact.

```
     module: ietf-sztp-conveyed-info

        yang-data conveyed-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 [RFC8259].

```
   {
     "ietf-sztp-conveyed-info:redirect-information" : {
       "bootstrap-server" : [
         {
           "address" : "sztp1.example.com",
           "port" : 8443,
```

```
          "trust-anchor" : "base64encodedvalue=="
        },
        {
          "address" : "sztp2.example.com",
          "port" : 8443,
          "trust-anchor" : "base64encodedvalue=="
        },
        {
          "address" : "sztp3.example.com",
          "port" : 8443,
          "trust-anchor" : "base64encodedvalue=="
        }
      ]
    }
  }
```

   The following example illustrates how onboarding information
   (Section 2.2) can be encoded using JSON [RFC8259].

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

```
  {
    "ietf-sztp-conveyed-info:onboarding-information" : {
      "boot-image" : {
        "os-name" : "VendorOS",
        "os-version" : "17.2R1.6",
        "download-uri" : [ "https://example.com/path/to/image/file" ],
        "image-verification" : [
          {
            "hash-algorithm" : "ietf-sztp-conveyed-info: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 conveyed 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 [RFC8040]; and
   an encoding defined in [ITU.X690.2015].

   <CODE BEGINS> file "ietf-sztp-conveyed-info@2019-04-30.yang"
   module ietf-sztp-conveyed-info {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info";
     prefix sztp-info;

     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-restconf {
       prefix rc;
       reference
         "RFC 8040: RESTCONF Protocol";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";
     contact
       "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
        WG List:  <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";
     description
       "This module defines the data model for the conveyed
        information artifact defined in RFC 8572 ('Secure Zero Touch
        Provisioning (SZTP)').

        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 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.

```
    revision 2019-04-30 {
      description
        "Initial version";
      reference
        "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
    }

    // 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
   rc:yang-data conveyed-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 of
            RFC 8572.  Its purpose is to redirect a device to
            another bootstrap server.";
         reference
           "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
         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 of
          RFC 8572.  Its purpose is to provide the device everything
          it needs to bootstrap itself.";
       reference
         "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
       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 (e.g., 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 (e.g., 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 content (e.g., XML) 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 a 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 SZTP 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-sztp-bootstrap-server

     rpcs:
       +---x get-bootstrapping-data
       │  +---w input
       │  │  +---w signed-data-preferred?   empty
       │  │  +---w hw-model?                string
       │  │  +---w os-name?                 string
       │  │  +---w os-version?              string
       │  │  +---w nonce?                   binary
       │  +--ro output
       │     +--ro reporting-level?    enumeration {onboarding-server}?
       │     +--ro conveyed-information    cms
       │     +--ro owner-certificate?      cms
       │     +--ro ownership-voucher?      cms
       +---x report-progress {onboarding-server}?
          +---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 (one to an untrusted bootstrap server and the other to a
   trusted bootstrap server), and one example is provided for the
   "report-progress" RPC.

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

   REQUEST

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

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

   <input
     xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-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 additional input parameters to the bootstrap server,
   which it may use when formulating its response to the device.


   REQUEST

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

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

   <input
     xmlns="urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server">
     <hw-model>model-x</hw-model>
     <os-name>vendor-os</os-name>
     <os-version>17.3R2.1</os-version>
     <nonce>extralongbase64encodedvalue=</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

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

POST /restconf/operations/ietf-sztp-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-sztp-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], and [RFC8366]; uses an encoding defined in
   [ITU.X690.2015]; and makes a reference to [RFC4250], [RFC6187], and
   [Std-802.1AR].

   <CODE BEGINS> file "ietf-sztp-bootstrap-server@2019-04-30.yang"
   module ietf-sztp-bootstrap-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server";
     prefix sztp-svr;

     organization
       "IETF NETCONF (Network Configuration) Working Group";
     contact
       "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
        WG List:  <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";
     description
       "This module defines an interface for bootstrap servers, as
        defined by RFC 8572 ('Secure Zero Touch Provisioning (SZTP)').

        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 (RFC 2119)
        (RFC 8174) 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
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC 8572; see the
        RFC itself for full legal notices.";

     revision 2019-04-30 {
       description

```
        "Initial version";
      reference
        "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
    }

    // features

    feature redirect-server {
      description
        "The server supports being a 'redirect server'.";
    }

    feature onboarding-server {
      description
        "The server supports being an 'onboarding server'.";
    }

    // 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 signed-data-preferred {
          type empty;
          description
            "This optional input parameter enables a device to
             communicate to the bootstrap server that it prefers
```

```
            to receive signed data.  Devices SHOULD always send
            this parameter when the bootstrap server is untrusted.
            Upon receiving this input parameter, the bootstrap
            server MUST return either signed data or unsigned
            redirect information; the bootstrap server MUST NOT
            return unsigned onboarding information.";
        }
        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.";
          reference
            "IEEE 802.1AR: 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 "16..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 {
          if-feature "onboarding-server";
          type enumeration {
            enum minimal {
              description
                "Send just the progress reports required by RFC 8572.";
              reference
                "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
            }
            enum verbose {
              description
                "Send additional progress reports that might help
                 troubleshooting an SZTP bootstrapping issue.";
            }
          }
          default "minimal";
          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
             '<signed-data-preferred>' input parameter).";
        }
        leaf conveyed-information {
          type cms;
          mandatory true;
          description
            "An SZTP conveyed information artifact, as described in
             Section 3.1 of RFC 8572.";
          reference
            "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
        }
        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 8572.  This leaf is optional because it is
             only needed when the conveyed information artifact is
             signed.";
          reference
            "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
        }
        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 8572.  This leaf is optional because it is
             only needed when the conveyed information artifact is
             signed.";
          reference
            "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
        }
      }
    }

    rpc report-progress {
      if-feature "onboarding-server";
      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
         bootstrap 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 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 bootstrapping attempt.";
          }
          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
             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
             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 and 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-para\\
                           meters>
                           ('\\' 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; that is:

                      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 topmost content type MUST be the
               signed-data content type, as described by Section 5 of
               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 last certificate in the chain
               MUST be the issuer for the device's end-entity
               certificate.

               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 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 SZTP Redirect Option

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

              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)   .
             .                                               .
             +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

             * option-code: OPTION_V4_SZTP_REDIRECT (143)
             * option-length: The option length in octets.
             * 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 SZTP Redirect Option

DHCPv4 Client Behavior

Clients MAY request the OPTION_V4_SZTP_REDIRECT option by including
its option code in the Parameter Request List (55) in DHCP request
messages.

On receipt of a DHCPv4 Reply message that contains the
OPTION_V4_SZTP_REDIRECT option, 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 the received OPTION_V4_SZTP_REDIRECT option 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 the decoding
agent behavior described in [RFC3396], to correctly process a URI
list split across a number of received OPTION_V4_SZTP_REDIRECT option
instances.

DHCPv4 Server Behavior

The DHCPv4 server MAY include a single instance of the
OPTION_V4_SZTP_REDIRECT option in DHCP messages it sends.  Servers
MUST NOT send more than one instance of the OPTION_V4_SZTP_REDIRECT
option.

The server's DHCP message MUST contain only a single instance of the
OPTION_V4_SZTP_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_SZTP_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_SZTP_REDIRECT MUST be split
into multiple instances of the option according to the process
described in [RFC3396].

8.2.  DHCPv6 SZTP Redirect Option

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

```
    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)            .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   * option-code: OPTION_V6_SZTP_REDIRECT (136)
   * option-length: The option length in octets.
   * 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 SZTP Redirect Option


   DHCPv6 Client Behavior

   Clients MAY request OPTION_V6_SZTP_REDIRECT using the process 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 that contains the
   OPTION_V6_SZTP_REDIRECT option, 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 the received OPTION_V6_SZTP_REDIRECT option 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.

The OPTION_V6_SZTP_REDIRECT option is a singleton.  Servers MUST NOT
send more than one instance of this 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 a DHCP
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             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...-+-+-+-+-+-+-+
```

   * uri-length: 2 octets long; specifies the length of the URI data.
   * URI: URI of the SZTP bootstrap server.

The URI of the SZTP bootstrap server MUST use the "https" URI scheme
defined in Section 2.7.2 of [RFC7230], and it 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 revocable.  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 that there is an IETF document
that focuses on securing NTP [NTS-NTP].

9.2.  Use of IDevID Certificates

   IDevID certificates, as defined in [Std-802.1AR], 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 SZTP bootstrapping
   data 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] 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 (e.g., 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; 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 not be 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", and "hw-model").  It is RECOMMENDED
that devices only pass the "signed-data-preferred" 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: through either transport-level security or
   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 "signed-data-preferred" input parameter to the
   "get-bootstrapping-data" RPC, may dynamically generate a response
   that is signed.

   Bootstrap server administrators are RECOMMENDED to follow best
   practices 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 SZTP 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 supposed 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 data that is either unsigned or
      signed but not encrypted may disclose information to unwanted
      parties (e.g., an administrator of the bootstrap server).  This is
      a privacy issue only, but it 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 point
      of hiding it from the administrators of the bootstrap server,
      which may be maintained by a third party.

9.11.  Validity Period for Conveyed Information

   The conveyed information artifact 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
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 be created for each signed artifact.
Not only does this enable a validity period to be specified, for each
artifact, but it also enables the validity of each artifact to be
revoked.

9.12.  Cascading Trust via Redirects

   Redirect information (Section 2.1), by design, instructs a
   bootstrapping device to initiate an HTTPS connection to the specified
   bootstrap servers.

   When the redirect information is trusted, the redirect information
   can encode a trust anchor certificate used by the device to

   authenticate the TLS end-entity certificate presented by each
   bootstrap server.

   As a result, any compromise in an interaction providing redirect
   information may result in compromise of all subsequent interactions.

9.13.  Possible Reuse of Private Keys

   This document describes two uses for secure device identity
   certificates.

   The primary use is for when the device authenticates itself to a
   bootstrap server, using its private key for TLS-level client-
   certificate-based authentication.

   A secondary use is for when the device needs to decrypt provided
   bootstrapping artifacts, using its private key to decrypt the data
   or, more precisely, per Section 6 of [RFC5652], decrypt a symmetric
   key used to decrypt the data.

   Section 3.4 of this document allows for the possibility that the same
   secure device identity certificate is utilized for both uses, as
   [Std-802.1AR] states that a DevID certificate MAY have the
   "keyEncipherment" KeyUsage bit, in addition to the "digitalSignature"
   KeyUsage bit, set.

   While it is understood that it is generally frowned upon to reuse
   private keys, this document views such reuse acceptable as there are
   not any known ways to cause a signature made in one context to be
   (mis)interpreted as valid in the other context.

9.14.  Non-issue with Encrypting Signed Artifacts

   This document specifies the encryption of signed objects, as opposed
   to the signing of encrypted objects, as might be expected given well-
   publicized oracle attacks (e.g., the padding oracle attack).

   This document does not view such attacks as feasible in the context
   of the solution because the decrypted text never leaves the device.

9.15.  The "ietf-sztp-conveyed-info" YANG Module

   The "ietf-sztp-conveyed-info" 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., a removable storage device), 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 content is 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-sztp-conveyed-info" module defines a top-level "choice"
statement that declares the content is 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).

9.16.  The "ietf-sztp-bootstrap-server" YANG Module

The "ietf-sztp-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 pre-configured 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

   IANA has registered two URIs in the "ns" subregistry of the "IETF XML
   Registry" [RFC3688] maintained at <https://www.iana.org/assignments/
   xml-registry>.  The following registrations have been made per the
   format in [RFC3688]:

      URI: urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info
      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-sztp-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

   IANA has registered two YANG modules in the "YANG Module Names"
   registry [RFC6020] maintained at <https://www.iana.org/assignments/
   yang-parameters>.  The following registrations have been made per the
   format in [RFC6020]:

      name:      ietf-sztp-conveyed-info
      namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-conveyed-info
      prefix:    sztp-info
      reference: RFC 8572

      name:      ietf-sztp-bootstrap-server
      namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-bootstrap-server
      prefix:    sztp-svr
      reference: RFC 8572

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

   IANA has registered two subordinate object identifiers in the "SMI
   Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)"
   registry maintained at <https://www.iana.org/assignments/
   smi-numbers>.  The following registrations have been made per the
   format in Section 3.4 of [RFC7107]:

      Decimal   Description                  References
      -------   --------------------------   ----------
      42        id-ct-sztpConveyedInfoXML    RFC 8572
      43        id-ct-sztpConveyedInfoJSON   RFC 8572

   id-ct-sztpConveyedInfoXML indicates that the "conveyed-information"
   is encoded using XML.  id-ct-sztpConveyedInfoJSON indicates that the
   "conveyed-information" is encoded using JSON.

10.4.  The BOOTP Vendor Extensions and DHCP Options Registry

   IANA has registered one DHCP code point in the "BOOTP Vendor
   Extensions and DHCP Options" registry maintained at
   <https://www.iana.org/assignments/bootp-dhcp-parameters>:

      Tag:         143
      Name:        OPTION_V4_SZTP_REDIRECT
      Data Length: N
      Meaning:     This option provides a list of URIs
                   for SZTP bootstrap servers
      Reference:   RFC 8572

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

   IANA has registered one DHCP code point in the "Option Codes"
   subregistry of the "Dynamic Host Configuration Protocol for IPv6
   (DHCPv6)" registry maintained at <https://www.iana.org/assignments/
   dhcpv6-parameters>:

      Value:            136
      Description:      OPTION_V6_SZTP_REDIRECT
      Client ORO:       Yes
      Singleton Option: Yes
      Reference:        RFC 8572

10.6.  The Service Name and Transport Protocol Port Number Registry

   IANA has registered 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>.  The
   following registration has been made per the format in Section 8.1.1
   of [RFC6335]:

      Service Name:             sztp
      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 "_sztp" for discovering
                                SZTP bootstrap servers.
      Reference:                RFC 8572
      Port Number:              N/A
      Service Code:             N/A
      Known Unauthorized Uses:  N/A
      Assignment Notes:         This protocol uses HTTPS as a substrate.

10.7.  The Underscored and Globally Scoped DNS Node Names Registry

   IANA has registered one service name in the "Underscored and Globally
   Scoped DNS Node Names" subregistry [RFC8552] of the "Domain Name
   System (DNS) Parameters" registry maintained at
   <https://www.iana.org/assignments/dns-parameters>.  The following
   registration has been made per the format in Section 3 of [RFC8552]:

       RR Type:            TXT
       _NODE NAME:         _sztp
       Reference:          RFC 8572

11.  References

11.1.  Normative References

   [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>.

   [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>.

   [RFC8552]  Crocker, D., "Scoped Interpretation of DNS Resource
              Records through "Underscored" Naming of Attribute Leaves",
              BCP 222, RFC 8552, DOI 10.17487/RFC8552, March 2019,
              <https://www.rfc-editor.org/info/rfc8552>.

   [Std-802.1AR]
              IEEE, "IEEE Standard for Local and metropolitan area
              networks - Secure Device Identity", IEEE 802.1AR.

11.2.  Informative References

   [NTS-NTP]  Franke, D., Sibold, D., Teichel, K., Dansarie, M., and
              R. Sundblad, "Network Time Security for the Network Time
              Protocol", Work in Progress, draft-ietf-ntp-using-nts-for-
              ntp-18, April 2019.

   [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>.

   [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>.

   [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>.

   [RFC8259]  Bray, T., Ed., "The JavaScript Object Notation (JSON) Data
              Interchange Format", STD 90, RFC 8259,
              DOI 10.17487/RFC8259, December 2017,
              <https://www.rfc-editor.org/info/rfc8259>.

   [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>.

   [YANG-CRYPTO-TYPES]
              Watsen, K. and H. Wang, "Common YANG Data Types for
              Cryptography", Work in Progress, draft-ietf-netconf-
              crypto-types-05, March 2019.

   [YANG-TRUST-ANCHORS]
              Watsen, K., "YANG Data Model for Global Trust Anchors",
              Work in Progress, draft-ietf-netconf-trust-anchors-03,
              March 2019.

Appendix A.  Example Device Data Model

   This section defines a non-normative data model that enables the
   configuration of SZTP bootstrapping and the 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 SZTP device
   data model.

```
 module: example-device-data-model
   +--rw sztp
      +--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-trust-anchors {bootstrap-servers}?
      |  +--ro reference*   ta:pinned-certificates-ref
      +--ro voucher-trust-anchors {signed-data}?
         +--ro reference*   ta:pinned-certificates-ref
```

   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 the device's factory default configuration and that it
   would be either set to "false" or deleted when the SZTP bootstrapping
   is longer needed.

A.2.  Example Usage

   Following is an instance example for this data model.

```
<sztp xmlns="https://example.com/sztp-device-data-model">
  <enabled>true</enabled>
  <idevid-certificate>base64encodedvalue==</idevid-certificate>
  <bootstrap-servers>
    <bootstrap-server>
      <address>sztp1.example.com</address>
      <port>8443</port>
    </bootstrap-server>
    <bootstrap-server>
      <address>sztp2.example.com</address>
      <port>8443</port>
    </bootstrap-server>
    <bootstrap-server>
      <address>sztp3.example.com</address>
      <port>8443</port>
    </bootstrap-server>
  </bootstrap-servers>
  <bootstrap-server-trust-anchors>
    <reference>manufacturers-root-ca-certs</reference>
  </bootstrap-server-trust-anchors>
  <voucher-trust-anchors>
    <reference>manufacturers-root-ca-certs</reference>
  </voucher-trust-anchors>
</sztp>
```

A.3.  YANG Module

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

   This module references [Std-802.1AR] and uses data types defined in
   [RFC6991], [YANG-CRYPTO-TYPES], and [YANG-TRUST-ANCHORS].

```
module example-device-data-model {
  yang-version 1.1;
  namespace "https://example.com/sztp-device-data-model";
  prefix sztp-ddm;

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

  import ietf-crypto-types {
```

```
      prefix ct;
      revision-date 2019-03-09;
      description
       "ietf-crypto-types is defined in
        draft-ietf-netconf-crypto-types";
      reference
       "draft-ietf-netconf-crypto-types-05:
          Common YANG Data Types for Cryptography";
    }

    import ietf-trust-anchors {
      prefix ta;
      revision-date 2019-03-09;
      description
       "ietf-trust-anchors is defined in
        draft-ietf-netconf-trust-anchors.";
      reference
       "draft-ietf-netconf-trust-anchors-03:
          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 SZTP
       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 2019-04-30 {
      description
        "Initial version";
      reference
        "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
    }

    // 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 sztp {
     description
       "Top-level container for the SZTP data model.";
     leaf enabled {
       type boolean;
       default false;
       description
         "The 'enabled' leaf controls if SZTP 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
          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:
            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.";
          }
        }
      }
      container bootstrap-server-trust-anchors {
        if-feature bootstrap-servers;
        config false;
        description "Container for a list of trust anchor references.";
        leaf-list reference {
          type ta:pinned-certificates-ref;
          description
            "A reference to a list of pinned certificate authority (CA)
             certificates that the device uses to validate bootstrap
             servers with.";
        }
      }
      container voucher-trust-anchors {
        if-feature signed-data;
        config false;
        description "Container for a list of trust anchor references.";
        leaf-list reference {
          type ta:pinned-certificates-ref;
          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 ("signed-data-preferred", 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
       "signed-data-preferred" input parameter, and optionally also the
       "nonce" input parameter, in the "get-bootstrapping-data" RPC.
       The "signed-data-preferred" 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 Manufacturer Authorized Signing
       Authority (MASA), which may be important for devices that do not
       have a reliable clock.

   2.  The bootstrap server, seeing the "signed-data-preferred" input
       parameter, knows that it can send either 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 MASA and download a voucher 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 "signed-data-preferred" 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 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 SZTP
   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 of 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 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 SZTP
   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    | +------+ +------+
      +---+   | Bootstrap | |  Bootstrap  | |Local | |Local | +---------+
      |NMS|   |  Server   | |   Server    | | DNS  | | DHCP | |Removable|
      +---+   +-----------+ +-------------+ |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, the manufacturer-hosted
      bootstrap server will either 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 support only redirect
      information or only onboarding information, while others may
      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 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., a 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.

```
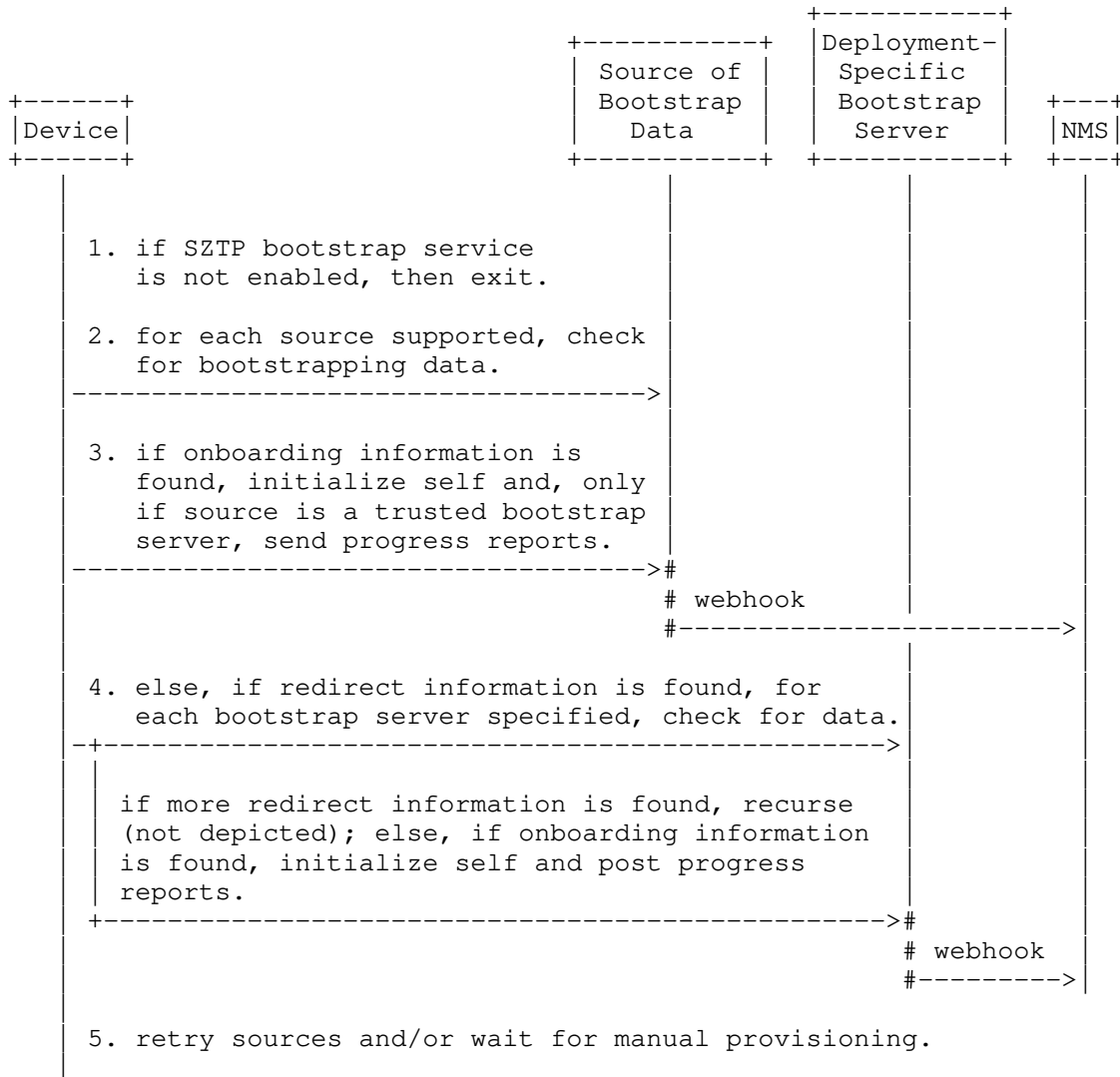                                            +-----------+
                             +-----------+  |Deployment-|
                             | Source of |  | Specific  |
      +------+               | Bootstrap |  | Bootstrap |  +---+
      |Device|               |   Data    |  |  Server   |  |NMS|
      +------+               +-----------+  +-----------+  +---+
         |                         |              |          |
         |                         |              |          |
         |   1. if SZTP bootstrap service         |          |
         |      is not enabled, then exit.        |          |
         |                         |              |          |
         |   2. for each source supported, check  |          |
         |      for bootstrapping data.           |          |
         |------------------------------------>   |          |
         |                         |              |          |
         |   3. if onboarding information is       |          |
         |      found, initialize self and, only  |          |
         |      if source is a trusted bootstrap  |          |
         |      server, send progress reports.    |          |
         |------------------------------------>#  |          |
         |                         |           # webhook     |
         |                         |           #------------------------>
         |                         |              |          |
         |   4. else, if redirect information is found, for   |
         |      each bootstrap server specified, check for data.
         |--+------------------------------------------------>|          |
         |  |                      |              |          |
         |  |  if more redirect information is found, recurse |
         |  |  (not depicted); else, if onboarding information |
         |  |  is 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 SZTP
       bootstrapping is configured, such as must be the case when
       running its "factory default" configuration.  If SZTP

       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.), should configure the device to either
          listen for NETCONF or RESTCONF connections or initiate call
          home connections [RFC8071], and should disable the SZTP
          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, 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 SZTP bootstrapping service, as the need for
        bootstrapping would no longer be present.

Acknowledgements

    The authors would like to thank the following for lively discussions
    on list and in the halls (ordered by last name): Michael Behringer,
    Martin Bjorklund, Dean Bogdanovic, Joe Clarke, Dave Crocker, Toerless
    Eckert, Stephen Farrell, Stephen Hanna, Wes Hardaker, David
    Harrington, Benjamin Kaduk, Radek Krejci, Suresh Krishnan, Mirja
    Kuehlewind, David Mandelberg, Alexey Melnikov, Russ Mundy, Reinaldo
    Penno, Randy Presuhn, Max Pritikin, Michael Richardson, Adam Roach,
    Juergen Schoenwaelder, and Phil Shafer.

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

Authors' Addresses

    Kent Watsen
    Watsen Networks

    Email: kent+ietf@watsen.net


    Ian Farrer
    Deutsche Telekom AG

    Email: ian.farrer@telekom.de


    Mikael Abrahamsson
    T-Systems

    Email: mikael.abrahamsson@t-systems.se