Workgroup: NETCONF Working Group
Internet-Draft:
draft-ietf-netconf-trust-anchors-20
Published: 12 December 2022
Intended Status: Standards Track
Expires: 15 June 2023
Authors: K. Watsen
Watsen Networks

A YANG Data Model for a Truststore

Abstract

This document defines a YANG module for configuring bags of certificates and bags of public keys that can be referenced by other data models for trust. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

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

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

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types

* BBBB --> the assigned RFC value for this draft

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

* 2022-12-12 --> the publication date of this draft

The "Relation to other RFCs" section Section 1.1 contains the text "one or more YANG modules" and, later, "modules". This text is sourced from a file in a context where it is unknown how many modules a draft defines. The text is not wrong as is, but it may be improved by stating more directly how many modules are defined.

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding Informative Reference in the Appendix.

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

* Appendix A. Change Log

Status of This Memo

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

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

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

This Internet-Draft will expire on 15 June 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction
  1.1. Relation to other RFCs
  1.2. Specification Language
  1.3. Adherence to the NMDA
  1.4. Conventions
2. The "ietf-truststore" Module
  2.1. Data Model Overview
  2.2. Example Usage
  2.3. YANG Module
3. Support for Built-in Trust Anchors

4. Security Considerations
4.1. Security of Data at Rest
4.2. Unconstrained Public Key Usage
4.3. The "ietf-truststore" YANG Module
5. IANA Considerations
5.1. The "IETF XML" Registry
5.2. The "YANG Module Names" Registry
6. References
6.1. Normative References
6.2. Informative References
Appendix A. Change Log
A.1. 00 to 01
A.2. 01 to 02
A.3. 02 to 03
A.4. 03 to 04
A.5. 04 to 05
A.6. 05 to 06
A.7. 06 to 07
A.8. 07 to 08
A.9. 08 to 09
A.10. 09 to 10
A.11. 10 to 11
A.12. 11 to 12
A.13. 12 to 13
A.14. 13 to 14
A.15. 14 to 15
A.16. 15 to 16
A.17. 16 to 17
A.18. 17 to 18
A.19. 18 to 19
A.20. 19 to 20
Acknowledgements
Author's Address

1. Introduction

This document defines a YANG 1.1 [RFC7950] module having the following characteristics:

Provide a central truststore for storing raw public keys and/or certificates.

Provide support for storing named bags of raw public keys and/or named bags of certificates.

Provide types that can be used to reference raw public keys or certificates stored in the central truststore.

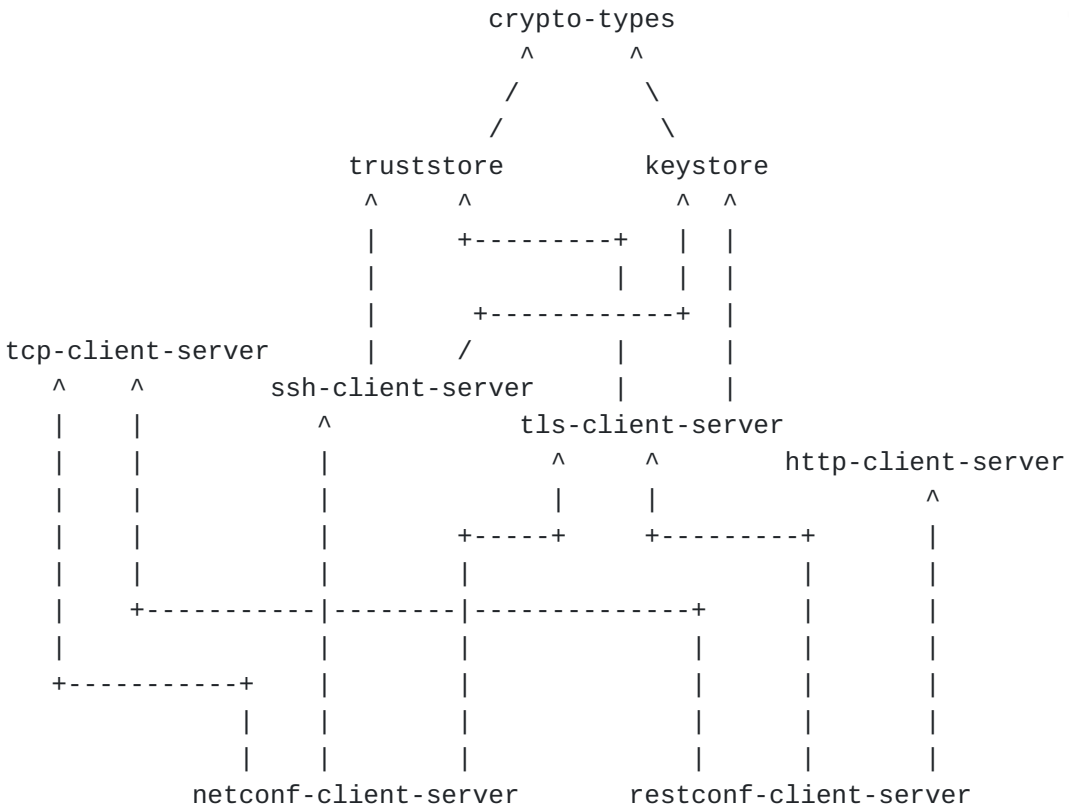
Provide groupings that enable raw public keys and certificates to be configured locally or as references truststore instances.

Enable the truststore to be instantiated in other data models, in addition to or in lieu of the central truststore instance.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of both the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The normative dependency relationship between the various RFCs in this collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]

tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification 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.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, trust anchors installed during manufacturing (e.g., for trusted well-known services), are expected to appear in <operational> (see Section 3).

1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-truststore" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-truststore". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-truststore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-truststore" module:

```
Features:
  +-- central-truststore-supported
  +-- local-definitions-supported
  +-- certificates
  +-- public-keys
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Typedefs

The following diagram lists the "typedef" statements defined in the "ietf-truststore" module:

```
Typedefs:
  leafref
    +-- certificate-bag-ref
    +-- certificate-ref
    +-- public-key-bag-ref
    +-- public-key-ref
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* All the typedefs defined in the "ietf-truststore" module extend the base "leafref" type defined in [RFC7950].

* The leafrefs refer to certificates, public keys, and bags in the central truststore, when this module is implemented.

* These typedefs are provided as an aid to downstream modules that import the "ietf-truststore" module.

2.1.3. Groupings

The "ietf-truststore" module defines the following "grouping" statements:

* local-or-truststore-certs-grouping
* local-or-truststore-public-keys-grouping
* truststore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "local-or-truststore-certs-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-truststore-certs-grouping" grouping:

```
grouping local-or-truststore-certs-grouping:
  +-- (local-or-truststore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +-- certificate* [name]
     |        +-- name?                            string
     |        +---u ct:trust-anchor-cert-grouping
     +--:(truststore) {central-truststore-supported,certificates}?
        +-- truststore-reference?   ts:certificate-bag-ref
```

Comments:

*The "local-or-truststore-certs-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option whether a bag of certificates can be defined locally or as a reference to a bag in the truststore.

*A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a bag in an alternate location.

*For the "local-definition" option, the "certificate" node uses the "trust-anchor-cert-grouping" grouping discussed in Section 2.1.4.7 of [I-D.ietf-netconf-crypto-types].

*For the "truststore" option, the "truststore-reference" is an instance of the "certificate-bag-ref" discussed in Section 2.1.2.

2.1.3.2. The "local-or-truststore-public-keys-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-truststore-public-keys-grouping" grouping:

```
grouping local-or-truststore-public-keys-grouping:
  +-- (local-or-truststore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +-- public-key* [name]
     |        +-- name?                      string
     |        +---u ct:public-key-grouping
     +--:(truststore) {central-truststore-supported,public-keys}?
        +-- truststore-reference?   ts:public-key-bag-ref
```

Comments:

*The "local-or-truststore-public-keys-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option whether a bag of public keys can be defined locally or as a reference to a bag in the truststore.

*A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a bag in an alternate location.

*For the "local-definition" option, the "public-key" node uses the "public-key-grouping" grouping discussed in Section 2.1.4.4 of [I-D.ietf-netconf-crypto-types].

*For the "truststore" option, the "truststore-reference" is an instance of the "certificate-bag-ref" discussed in Section 2.1.2.

2.1.3.3. The "truststore-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "truststore-grouping" grouping:

```
grouping truststore-grouping:
  +-- certificate-bags {certificates}?
  |  +-- certificate-bag* [name]
  |     +-- name?          string
  |     +-- description?   string
  |     +-- certificate* [name]
  |        +-- name?                            string
  |        +---u ct:trust-anchor-cert-grouping
  +-- public-key-bags {public-keys}?
     +-- public-key-bag* [name]
        +-- name?          string
        +-- description?   string
        +-- public-key* [name]
           +-- name?                     string
           +---u ct:public-key-grouping
```

Comments:

*The "truststore-grouping" grouping defines a truststore instance as being composed of certificates and/or public keys, both of which are enabled by "feature" statements. The structure supporting certificates and public keys is essentially the same, having an outer list of "bags" containing in inner list of objects (certificates or public keys). The bags enable trust anchors serving a common purpose to be grouped and referenced together.

*For certificates, each certificate is defined by the "trust-anchor-cert-grouping" grouping Section 2.1.4.7 of [I-D.ietf-netconf-crypto-types]. Thus the "cert-data" node is a CMS structure that can be composed of a chain of one or more certificates. Additionally, the "certificate-expiration"

notification enables the server to alert clients when certificates are nearing or have already expired.

*For public keys, each public key is defined by the "public-key-grouping" grouping Section 2.1.4.4 of [I-D.ietf-netconf-crypto-types]. Thus the "public-key" node can be one of any number of structures specified by the "public-key-format" identity node.

2.1.4. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-truststore" module, without expanding the "grouping" statements:

```
module: ietf-truststore
  +--rw truststore {central-truststore-supported}?
     +---u truststore-grouping
```

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-truststore" module, with all "grouping" statements expanded, enabling the truststore's full structure to be seen:

```
module: ietf-truststore
  +--rw truststore {central-truststore-supported}?
     +--rw certificate-bags {certificates}?
     |  +--rw certificate-bag* [name]
     |     +--rw name           string
     |     +--rw description?   string
     |     +--rw certificate* [name]
     |        +--rw name                      string
     |        +--rw cert-data                 trust-anchor-cert-cms
     |        +---n certificate-expiration
     |                {certificate-expiration-notification}?
     |           +-- expiration-date    yang:date-and-time
     +--rw public-key-bags {public-keys}?
        +--rw public-key-bag* [name]
           +--rw name           string
           +--rw description?   string
           +--rw public-key* [name]
              +--rw name                string
              +--rw public-key-format   identityref
              +--rw public-key          binary
```

Comments:

*Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

*The protcol-accessible nodes for the "ietf-truststore" module are an instance of the "truststore-grouping" grouping discussed in Section 2.1.3.3.

*The top-level node "truststore" is additionally constrained by the feature "central-truststore-supported".

*The "truststore-grouping" grouping is discussed in Section 2.1.3.3.

*The reason for why the "truststore-grouping" exists separate from the protocol-accessible nodes definition is to enable instances of the truststore to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The examples in this section are encoded using XML, such as might be the case when using the NETCONF protocol. Other encodings MAY be used, such as JSON when using the RESTCONF protocol.

2.2.1. A Truststore Instance

This section presents an example illustrating trust anchors in <intended>, as per Section 2.1.4. Please see Section 3 for an example illustrating built-in values in <operational>.

The example contained in this section defines eight bags of trust anchors. There are four certificate-based bags and four public key based bags. The following diagram provides an overview of the contents in the example:

```
Certificate Bags
  +-- Trust anchor certs for authenticating a set of remote servers
  +-- End entity certs for authenticating a set of remote servers
  +-- Trust anchor certs for authenticating a set of remote clients
  +-- End entity certs for authenticating a set of remote clients

Public Key Bags
  +-- SSH keys to authenticate a set of remote SSH server
  +-- SSH keys to authenticate a set of remote SSH clients
  +-- Raw public keys to authenticate a set of remote SSH server
  +-- Raw public keys to authenticate a set of remote SSH clients
```

Following is the full example:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<truststore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- A bag of Certificate Bags -->
  <certificate-bags>

    <!-- Trust Anchor Certs for Authenticating Servers -->
    <certificate-bag>
      <name>trusted-server-ca-certs</name>
      <description>
        Trust anchors (i.e. CA certs) used to authenticate server
        certificates.  A server certificate is authenticated if its
        end-entity certificate has a chain of trust to one of these
        certificates.
      </description>
      <certificate>
        <name>Server Cert Issuer #1</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
      <certificate>
        <name>Server Cert Issuer #2</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

    <!-- End Entity Certs for Authenticating Servers -->
    <certificate-bag>
      <name>trusted-server-ee-certs</name>
      <description>
        Specific end-entity certificates used to authenticate server
        certificates.  A server certificate is authenticated if its
        end-entity certificate is an exact match to one of these
        certificates.
      </description>
      <certificate>
        <name>My Application #1</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
      <certificate>
        <name>My Application #2</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

    <!-- Trust Anchor Certs for Authenticating Clients -->
    <certificate-bag>
```

```
    <name>trusted-client-ca-certs</name>
    <description>
      Trust anchors (i.e. CA certs) used to authenticate client
      certificates.  A client certificate is authenticated if its
      end-entity certificate has a chain of trust to one of these
      certificates.
    </description>
    <certificate>
      <name>Client Identity Issuer #1</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
    <certificate>
      <name>Client Identity Issuer #2</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
  </certificate-bag>

  <!-- End Entity Certs for Authenticating Clients -->
  <certificate-bag>
    <name>trusted-client-ee-certs</name>
    <description>
      Specific end-entity certificates used to authenticate client
      certificates.  A client certificate is authenticated if its
      end-entity certificate is an exact match to one of these
      certificates.
    </description>
    <certificate>
      <name>George Jetson</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
    <certificate>
      <name>Fred Flintstone</name>
      <cert-data>BASE64VALUE=</cert-data>
    </certificate>
  </certificate-bag>
</certificate-bags>

<!-- A List of Public Key Bags -->
<public-key-bags>

  <!-- Public Keys for Authenticating SSH Servers -->
  <public-key-bag>
    <name>trusted-ssh-public-keys</name>
    <description>
      Specific SSH public keys used to authenticate SSH server
      public keys.  An SSH server public key is authenticated if
      its public key is an exact match to one of these public keys.

      This list of SSH public keys is analogous to OpenSSH's
```

```
        "/etc/ssh/ssh_known_hosts" file.
      </description>
      <public-key>
        <name>corp-fw1</name>
        <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
      <public-key>
        <name>corp-fw2</name>
        <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </public-key-bag>

    <!-- SSH Public Keys for Authenticating Application A -->
    <public-key-bag>
      <name>SSH Public Keys for Application A</name>
      <description>
        SSH public keys used to authenticate application A's SSH
        public keys.  An SSH public key is authenticated if it
        is an exact match to one of these public keys.
      </description>
      <public-key>
        <name>Application Instance #1</name>
        <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
      <public-key>
        <name>Application Instance #2</name>
        <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </public-key-bag>

    <!-- Raw Public Keys for TLS Servers -->
    <public-key-bag>
      <name>Raw Public Keys for TLS Servers</name>
      <public-key>
        <name>Raw Public Key #1</name>
        <public-key-format>ct:subject-public-key-info-format</public\
-key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
      <public-key>
        <name>Raw Public Key #2</name>
```

```
        <public-key-format>ct:subject-public-key-info-format</public\
-key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </public-key-bag>

    <!-- Raw Public Keys for TLS Clients -->
    <public-key-bag>
      <name>Raw Public Keys for TLS Clients</name>
      <public-key>
        <name>Raw Public Key #1</name>
        <public-key-format>ct:subject-public-key-info-format</public\
-key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
      <public-key>
        <name>Raw Public Key #2</name>
        <public-key-format>ct:subject-public-key-info-format</public\
-key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </public-key-bag>
  </public-key-bags>
</truststore>
```

2.2.2. A Certificate Expiration Notification

The following example illustrates the "certificate-expiration" notification (per Section 2.1.4.6 of [I-D.ietf-netconf-crypto-types]) for a certificate configured in the truststore in Section 2.2.1.

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
    <certificate-bags>
      <certificate-bag>
        <name>trusted-client-ee-certs</name>
        <certificate>
          <name>George Jetson</name>
          <certificate-expiration>
            <expiration-date>2018-08-05T14:18:53-05:00</expiration-d\
ate>
          </certificate-expiration>
        </certificate>
      </certificate-bag>
    </certificate-bags>
  </truststore>
</notification>
```

2.2.3. The "Local or Truststore" Groupings

This section illustrates the various "local-or-truststore" groupings defined in the "ietf-truststore" module, specifically the "local-or-truststore-certs-grouping" (Section 2.1.3.1) and "local-or-truststore-public-keys-grouping" (Section 2.1.3.2) groupings.

These examples assume the existence of an example module called "ex-truststore-usage" having the namespace "http://example.com/ns/example-truststore-usage".

The ex-truststore-usage module is first presented using tree diagrams [RFC8340], followed by an instance example illustrating all the "local-or-truststore" groupings in use, followed by the YANG module itself.

The following tree diagram illustrates "ex-truststore-usage" without expanding the "grouping" statements:

```
module: ex-truststore-usage
  +--rw truststore-usage
     +--rw cert* [name]
     |  +--rw name                                    string
     |  +---u ts:local-or-truststore-certs-grouping
     +--rw public-key* [name]
        +--rw name                                          string
        +---u ts:local-or-truststore-public-keys-grouping
```

The following tree diagram illustrates the "ex-truststore-usage" module, with all "grouping" statements expanded, enabling the truststore's full structure to be seen:

```
module: ex-truststore-usage
  +--rw truststore-usage
     +--rw cert* [name]
     |  +--rw name                          string
     |  +--rw (local-or-truststore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw certificate* [name]
     |     |        +--rw name                     string
     |     |        +--rw cert-data
     |     |        |       trust-anchor-cert-cms
     |     |        +---n certificate-expiration
     |     |                {certificate-expiration-notification}?
     |     |           +-- expiration-date    yang:date-and-time
     |     +--:(truststore)
     |              {central-truststore-supported,certificates}?
     |        +--rw truststore-reference?   ts:certificate-bag-ref
     +--rw public-key* [name]
        +--rw name                          string
        +--rw (local-or-truststore)
           +--:(local) {local-definitions-supported}?
           |  +--rw local-definition
           |     +--rw public-key* [name]
           |        +--rw name                 string
           |        +--rw public-key-format    identityref
           |        +--rw public-key           binary
           +--:(truststore)
                    {central-truststore-supported,public-keys}?
              +--rw truststore-reference?   ts:public-key-bag-ref
```

The following example provides two equivalent instances of each grouping, the first being a reference to a truststore and the second being locally-defined. The instance having a reference to a truststore is consistent with the truststore defined in Section 2.2.1. The two instances are equivalent, as the locally-

defined instance example contains the same values defined by the truststore instance referenced by its sibling example.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<truststore-usage
  xmlns="http://example.com/ns/example-truststore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- The following two equivalent examples illustrate   -->
  <!-- the "local-or-truststore-certs-grouping" grouping: -->

  <cert>
    <name>example 1a</name>
    <truststore-reference>trusted-client-ca-certs</truststore-refere\
nce>
  </cert>

  <cert>
    <name>example 1b</name>
    <local-definition>
      <name>my-trusted-client-ca-certs</name>
      <certificate>
        <name>Client Identity Issuer #1</name>
        <cert>BASE64VALUE=</cert>
      </certificate>
      <certificate>
        <name>Client Identity Issuer #2</name>
        <cert>BASE64VALUE=</cert>
      </certificate>
    </local-definition>
  </cert>


  <!-- The following two equivalent examples illustrate the -->
  <!-- "local-or-truststore-public-keys-grouping" grouping: -->

  <public-key>
    <name>example 2a</name>
    <truststore-reference>trusted-ssh-public-keys</truststore-refere\
nce>
  </public-key>

  <public-key>
    <name>example 2b</name>
    <local-definition>
      <name>trusted-ssh-public-keys</name>
      <public-key>
        <name>corp-fw1</name>
        <public-key-format>
          ct:ssh-public-key-format
        </public-key-format>
        <public-key>BASE64VALUE=</public-key>
```

```
      </public-key>
      <public-key>
        <name>corp-fw2</name>
        <public-key-format>
          ct:ssh-public-key-format
        </public-key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </local-definition>
  </public-key>

</truststore-usage>
```

Following is the "ex-truststore-usage" module's YANG definition:

```
module ex-truststore-usage {
  yang-version 1.1;
  namespace "http://example.com/ns/example-truststore-usage";
  prefix etu;

  import ietf-truststore {
    prefix ts;
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  organization
    "Example Corporation";

  contact
    "Author: YANG Designer <mailto:yang.designer@example.com>";

  description
    "This module illustrates notable groupings defined in
     the 'ietf-truststore' module.";

  revision 2022-12-12 {
    description
      "Initial version";
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  container truststore-usage {
    description
      "An illustration of the various truststore groupings.";
    list cert {
      key "name";
      leaf name {
        type string;
        description
          "An arbitrary name for this cert.";
      }
      uses ts:local-or-truststore-certs-grouping;
      description
        "An cert that may be configured locally or be
         a reference to a cert in the truststore.";
    }
    list public-key {
      key "name";
      leaf name {
        type string;
        description
          "An arbitrary name for this cert.";
```

```
      }
      uses ts:local-or-truststore-public-keys-grouping;
      description
        "An public key that may be configured locally or be
         a reference to a public key in the truststore.";
    }
  }
}
```

2.3. YANG Module

This YANG module imports modules from [RFC8341] and [I-D.ietf-netconf-crypto-types].

<CODE BEGINS> file "ietf-truststore@2022-12-12.yang"

```
module ietf-truststore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-truststore";
  prefix ts;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

  contact
    "WG Web  : https://datatracker.ietf.org/wg/netconf
     WG List : NETCONF WG list <mailto:netconf@ietf.org>
     Author  : Kent Watsen <kent+ietf@watsen.net>";

  description
    "This module defines a 'truststore' to centralize management
     of trust anchors including certificates and public keys.

     Copyright (c) 2022 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 Revised
     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 BBBB
     (https://www.rfc-editor.org/info/rfcBBBB); see the RFC
     itself for full legal notices.

     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 2022-12-12 {
  description
    "Initial version";
  reference
    "RFC BBBB: A YANG Data Model for a Truststore";
}

/****************/
/*   Features   */
/****************/

feature central-truststore-supported {
  description
    "The 'central-truststore-supported' feature indicates that
     the server supports the truststore (i.e., implements the
     'ietf-truststore' module).";
}

feature local-definitions-supported {
  description
    "The 'local-definitions-supported' feature indicates that
     the server supports locally-defined trust anchors.";
}

feature certificates {
  description
    "The 'certificates' feature indicates that the server
     implements the /truststore/certificate-bags subtree.";
}

feature public-keys {
  description
    "The 'public-keys' feature indicates that the server
     implements the /truststore/public-key-bags subtree.";
}

/****************/
/*   Typedefs   */
/****************/

typedef certificate-bag-ref {
  type leafref {
    path "/ts:truststore/ts:certificate-bags/"
       + "ts:certificate-bag/ts:name";
  }
  description
    "This typedef defines a reference to a certificate bag
     in the truststore, when this module is implemented.";
}
```

```
typedef certificate-ref {
  type leafref {
    path "/ts:truststore/ts:certificate-bags/ts:certificate-bag"
       + "[ts:name = current()/../ts:certificate-bag]/"
       + "ts:certificate/ts:name";
  }
  description
    "This typedef defines a reference to a specific certificate
     in a certificate bag in the truststore, when this module
     is implemented.  This typedef requires that there exist a
     sibling 'leaf' node called 'certificate-bag' that SHOULD
     have the typedef 'certificate-bag-ref'.";
}

typedef public-key-bag-ref {
  type leafref {
    path "/ts:truststore/ts:public-key-bags/"
       + "ts:public-key-bag/ts:name";
  }
  description
    "This typedef defines a reference to a public key bag
     in the truststore, when this module is implemented.";
}

typedef public-key-ref {
  type leafref {
    path "/ts:truststore/ts:public-key-bags/ts:public-key-bag"
       + "[ts:name = current()/../ts:public-key-bag]/"
       + "ts:public-key/ts:name";
  }
  description
    "This typedef defines a reference to a specific public key
     in a public key bag in the truststore, when this module is
     implemented.  This typedef requires that there exist a
     sibling 'leaf' node called 'public-key-bag' that SHOULD
     have the typedef 'public-key-bag-ref'.";
}

/*****************/
/*   Groupings   */
/*****************/

grouping local-or-truststore-certs-grouping {
  description
    "A grouping that allows the certificates to be either
     configured locally, within the using data model, or be a
     reference to a certificate bag stored in the truststore.

     Servers that do not 'implement' this module, and hence
```

```
     'central-truststore-supported' is not defined, SHOULD
     augment in custom 'case' statements enabling references
     to the alternate truststore locations.";
  choice local-or-truststore {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice between an inlined definition and a definition
       that exists in the truststore.";
    case local {
      if-feature "local-definitions-supported";
      container local-definition {
        description
          "A container for locally configured trust anchor
           certificates.";
        list certificate {
          key "name";
          min-elements 1;
          description
            "A trust anchor certificate.";
          leaf name {
            type string;
            description
              "An arbitrary name for this certificate.";
          }
          uses ct:trust-anchor-cert-grouping {
            refine "cert-data" {
              mandatory true;
            }
          }
        }
      }
    }
    case truststore {
      if-feature "central-truststore-supported";
      if-feature "certificates";
      leaf truststore-reference {
        type ts:certificate-bag-ref;
        description
          "A reference to a certificate bag that exists in the
           truststore, when this module is implemented.";
      }
    }
  }
}

grouping local-or-truststore-public-keys-grouping {
  description
    "A grouping that allows the public keys to be either
```

```
     configured locally, within the using data model, or be a
     reference to a public key bag stored in the truststore.

     Servers that do not 'implement' this module, and hence
     'central-truststore-supported' is not defined, SHOULD
     augment in custom 'case' statements enabling references
     to the alternate truststore locations.";
  choice local-or-truststore {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice between an inlined definition and a definition
       that exists in the truststore.";
    case local {
      if-feature "local-definitions-supported";
      container local-definition {
        description
          "A container to hold local public key definitions.";
        list public-key {
          key "name";
          description
            "A public key definition.";
          leaf name {
            type string;
            description
              "An arbitrary name for this public key.";
          }
          uses ct:public-key-grouping;
        }
      }
    }
    case truststore {
      if-feature "central-truststore-supported";
      if-feature "public-keys";
      leaf truststore-reference {
        type ts:public-key-bag-ref;
        description
          "A reference to a bag of public keys that exists
           in the truststore, when this module is implemented.";
      }
    }
  }
}

grouping truststore-grouping {
  description
    "A grouping definition that enables use in other contexts.
     Where used, implementations MUST augment new 'case'
     statements into the various local-or-truststore 'choice'
```

```
   statements to supply leafrefs to the model-specific
   location(s).";
container certificate-bags {
  nacm:default-deny-write;
  if-feature "certificates";
  description
    "A collection of certificate bags.";
  list certificate-bag {
    key "name";
    description
      "A bag of certificates.  Each bag of certificates SHOULD
       be for a specific purpose.  For instance, one bag could
       be used to authenticate a specific set of servers, while
       another could be used to authenticate a specific set of
       clients.";
    leaf name {
      type string;
      description
        "An arbitrary name for this bag of certificates.";
    }
    leaf description {
      type string;
      description
        "A description for this bag of certificates.  The
         intended purpose for the bag SHOULD be described.";
    }
    list certificate {
      key "name";
      description
        "A trust anchor certificate.";
      leaf name {
        type string;
        description
          "An arbitrary name for this certificate.";
      }
      uses ct:trust-anchor-cert-grouping {
        refine "cert-data" {
          mandatory true;
        }
      }
    }
  }
}
container public-key-bags {
  nacm:default-deny-write;
  if-feature "public-keys";
  description
    "A collection of public key bags.";
  list public-key-bag {
```

```
        key "name";
        description
          "A bag of public keys.  Each bag of keys SHOULD be for
           a specific purpose.  For instance, one bag could be used
           authenticate a specific set of servers, while another
           could be used to authenticate a specific set of clients.";
        leaf name {
          type string;
          description
            "An arbitrary name for this bag of public keys.";
        }
        leaf description {
          type string;
          description
            "A description for this bag public keys.  The
             intended purpose for the bag SHOULD be described.";
        }
        list public-key {
          key "name";
          description
            "A public key.";
          leaf name {
            type string;
            description
              "An arbitrary name for this public key.";
          }
          uses ct:public-key-grouping;
        }
      }
    }
  }

  /*********************************/
  /*   Protocol accessible nodes   */
  /*********************************/

  container truststore {
    if-feature central-truststore-supported;
    nacm:default-deny-write;
    description
      "The truststore contains bags of certificates and
       public keys.";
    uses truststore-grouping;
  }
}
```

<CODE ENDS>

3. Support for Built-in Trust Anchors

In some implementations, a server may define some built-in trust anchors. For instance, there may be built-in trust anchors enabling the server to securely connect to well-known services (e.g., an SZTP [RFC8572] bootstrap server) or public CA certificates to connect to arbitrary services using public PKI.

Built-in trust anchors are expected to be set by a vendor-specific process. Any ability for operators to modify built-in trust anchors is outside the scope of this document.

As built-in trust anchors are provided by the server, they are present in <operational> (and <system> [I-D.ma-netmod-with-system], if used). The example below illustrates what the truststore in <operational> might look like for a server in its factory default state.

```
<truststore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <certificate-bags>

    <certificate-bag or:origin="or:system">
      <name>Built-In Manufacturer Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        manufacturer-signed objects, such as TLS server certificates,
        vouchers, etc.
      </description>
      <certificate>
        <name>Manufacturer Root CA Cert</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

    <certificate-bag or:origin="or:system">
      <name>Built-In Public Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        certificates issued by public certificate authorities,
        such as the end-entity certificate for web servers.
      </description>
      <certificate>
        <name>Public Root CA Cert 1</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
      <certificate>
        <name>Public Root CA Cert 2</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
      <certificate>
        <name>Public Root CA Cert 3</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

  </certificate-bags>
</truststore>
```

In order for the built-in bags of trust anchors and/or their trust anchors to be referenced by configuration, they MUST first be copied into <running>.

The built-in bags and/or their trust anchors MUST be copied into <running> using the same "key" values if it is desired for the server to maintain/update them (e.g., a software update may update a bag of trusted public CA certificates used for TLS-client connections).

Built-in bags and/or their trust anchors MAY be copied into other parts of the configuration but, by doing so, they lose their association to the built-in entries and any assurances afforded by knowing they are/were built-in.

The built-in bags and/or their trust anchors are immutable by configuration operations. Servers MUST ignore attempts to modify any aspect of built-in bags and/or their trust anchors from <running>.

The following example illustrates how a single built-in public CA certificate from the previous example has been propagated to <running>:

```
<truststore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <certificate-bags>

    <certificate-bag>
      <name>Built-In Public Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        certificates issued by public certificate authorities,
        such as the end-entity certificate for web servers.

        Only the subset of the certificates that are referenced
        by other configuration nodes need to be copied.  For
        instance, only "Public Root CA Cert 3" is present here.

        No new certificates can be added, nor existing certificate
        values changed.  Missing certificates have no effect on
        "operational" when the configuration is applied.
      </description>
      <certificate>
        <name>Public Root CA Cert 3</name>
        <cert-data>BASE64VALUE=</cert-data>
      </certificate>
    </certificate-bag>

  </certificate-bags>
</truststore>
```

4. Security Considerations

4.1. Security of Data at Rest

The YANG module defined in this document defines a mechanism called a "truststore" that, by its name, suggests that its contents are protected from unauthorized modification.

Security controls for the API (i.e., data in motion) are discussed in Section 4.3, but controls for the data at rest (e.g., on disk) cannot be specified by the YANG module.

In order to satisfy the expectations of a "truststore", it is RECOMMENDED that implementations ensure that the truststore contents are protected from unauthorized modifications when at rest.

4.2. Unconstrained Public Key Usage

This module enables the configuration of public keys without constraints on their usage, e.g., what operations the key is allowed to be used for (encryption, verification, both).

Trust anchors configured via this module are implicitly trusted to validate certification paths that may include any name, be used for any purpose and etc., subject to constraints imposed by an intermediate CA or by context in which the truststore is used. Implementations are free to use alternative or auxiliary structures and validation rules to define constraints that limit the applicability of any trust anchor.

4.3. The "ietf-truststore" YANG Module

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The Network 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.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

All the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "truststore" instance, may be considered sensitive or vulnerable in

some network environments. For instance, any modification to a trust anchor or reference to a trust anchor may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any "rpc" or "action" statements, and thus the security considerations for such is not provided here.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-truststore
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

5.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registration is requested:

```
name:         ietf-truststore
namespace:    urn:ietf:params:xml:ns:yang:ietf-truststore
prefix:       ts
reference:    RFC BBBB
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-25, 19 October 2022, <https://www.ietf.org/archive/id/draft-ietf-netconf-crypto-types-25.txt>.

[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>.

[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>.

[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>.

[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>.

6.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-11, 19 October 2022, <https://www.ietf.org/archive/id/draft-ietf-netconf-http-client-server-11.txt>.

[I-D.ietf-netconf-keystore] Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-26, 19 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-26>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-27, 19 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-27>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-27, 19 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-27>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-31, 19 October 2022, <https://

datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-31>.

[I-D.ietf-netconf-tcp-client-server] Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-14, 19 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-14>.

[I-D.ietf-netconf-tls-client-server] Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-31, 19 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-31>.

[I-D.ietf-netconf-trust-anchors] Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-19, 19 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-19>.

[I-D.ma-netmod-with-system] Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ma-netmod-with-system-05, 29 September 2022, <https://datatracker.ietf.org/doc/html/draft-ma-netmod-with-system-05>.

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

[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>.

[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>.

[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>.

[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>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Added features "x509-certificates" and "ssh-host-keys".

* Added nacm:default-deny-write to "trust-anchors" container.

A.2. 01 to 02

* Switched "list pinned-certificate" to use the "trust-anchor-cert-grouping" from crypto-types. Effectively the same definition as before.

A.3. 02 to 03

* Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

A.4. 03 to 04

* Added groupings 'local-or-truststore-certs-grouping' and 'local-or-truststore-host-keys-grouping', matching similar definitions in the keystore draft. Note new (and incomplete) "truststore" usage!

*Related to above, also added features 'truststore-supported' and 'local-trust-anchors-supported'.

A.5. 04 to 05

*Renamed "trust-anchors" to "truststore"

*Removed "pinned." prefix everywhere, to match truststore rename

*Moved everything under a top-level 'grouping' to enable use in other contexts.

*Renamed feature from 'local-trust-anchors-supported' to 'local-definitions-supported' (same name used in keystore)

Removed the "require-instance false" statement from the "-ref" typedefs.

*Added missing "ssh-host-keys" and "x509-certificates" if-feature statements

A.6. 05 to 06

*Editorial changes only.

A.7. 06 to 07

*Added Henk Birkholz as a co-author (thanks Henk!)

*Added PSKs and raw public keys to truststore.

A.8. 07 to 08

*Added new "Support for Built-in Trust Anchors" section.

*Removed spurious "uses ct:trust-anchor-certs-grouping" line.

*Removed PSK from model.

A.9. 08 to 09

*Removed remaining PSK references from text.

*Wrapped each top-level list with a container.

*Introduced "bag" term.

*Merged "SSH Public Keys" and "Raw Public Keys" in a single "Public Keys" bag. Consuming downstream modules (i.e., "ietf-[ssh/tls]-[client/server]) refine the "public-key-format" to be either SSH or TLS specific as needed.

A.10. 09 to 10

* Removed "algorithm" node from examples.

* Removed the no longer used statements supporting the old "ssh-public-key" and "raw-public-key" nodes.

* Added a "Note to Reviewers" note to first page.

A.11. 10 to 11

* Corrected module prefix registered in the IANA Considerations section.

* Modified 'local-or-truststore-certs-grouping' to use a list (not a leaf-list).

* Added new example section "The Local or Truststore Groupings".

* Clarified expected behavior for "built-in" certificates in <operational>

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.12. 11 to 12

* Fixed a copy/paste issue in the "Data at Rest" Security Considerations section.

A.13. 12 to 13

* Fixed issues found by the SecDir review of the "keystore" draft.

A.14. 13 to 14

* Added an "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir.

* Addressed comments raised by YANG Doctor.

A.15. 14 to 15

* Added prefixes to 'path' statements per trust-anchors/issues/1

* Renamed feature "truststore-supported" to "central-truststore-supported".

*Associated with above, generally moved text to refer to a "central" truststore.

*Removed two unecessary/unwanted "min-elements 1" and associated "presence" statements.

*Aligned modules with `pyang -f` formatting.

*Fixed nits found by YANG Doctor reviews.

A.16. 15 to 16

*Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

*Minor editorial nits

A.17. 16 to 17

*fixup the 'WG Web' and 'WG List' lines in YANG module(s)

*fixup copyright (i.e., s/Simplified/Revised/) in YANG module(s)

*Added Informative reference to ma-netmod-with-system

A.18. 17 to 18

*Updated Security Considerations section to address comment received from Carl Wallace.

*Fixed examples to not have line-returns around "identity" encodings.

*Fixed a couple tree diagrams to not create diagrams for "groupings" too.

*Added "if-feature central-truststore-supported" to top-level "trustore" container.

A.19. 18 to 19

*Updated per Shepherd reviews impacting the suite of drafts.

A.20. 19 to 20

*Updated per Shepherd reviews impacting the suite of drafts.

Acknowledgements

The authors especially thank Henk Birkholz for contributing YANG to the ietf-truststore module supporting raw public keys and PSKs (pre-shared or pairwise-symmetric keys). While these contributions were

eventually replaced by reusing the existing support for asymmetric and symmetric trust anchors, respectively, it was only thru Henk's initiative that the WG was able to come to that result.

The authors additionally thank the following for helping give shape to this work (ordered by first name): Balazs Kovacs, Carl Wallace, Eric Voit, Juergen Schoenwaelder, Liang Xia, Martin Bjoerklund, Nick Hancock, Qin Wu, and Yoav Nir.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net