NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 11 January 2021

K. Watsen
Watsen Networks

10 July 2020

A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-12

Abstract

This document defines a YANG 1.1 data model for configuring globally-accessible bags of certificates and public keys that can be referenced by other data models for trust.

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:

* "2020-07-10" --> the publication date of this draft

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/.

Copyright Notice

Table of Contents

1. Introduction

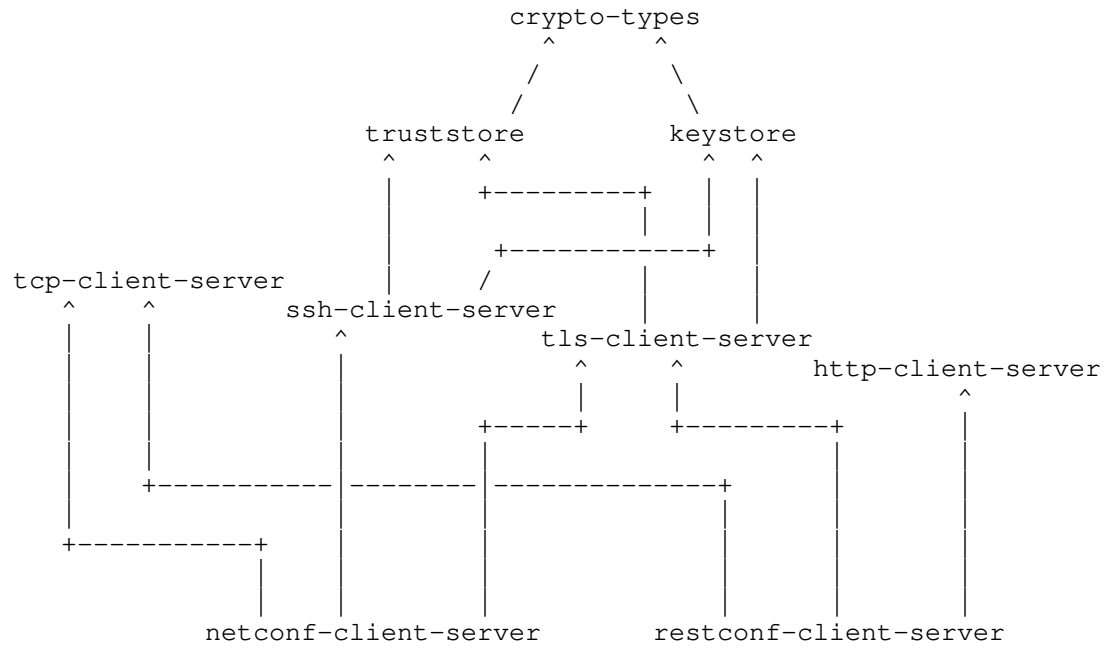
This document defines a YANG 1.1 [RFC7950] data model for configuring globally-accessible bags of certificates and public keys that can be referenced by other data models for trust.

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 define configuration modules for clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Links the 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 in 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).

2. The "ietf-truststore" Module

This section defines a YANG 1.1 [RFC7950] module that defines a "truststore" and groupings supporting downstream modules to reference the truststore or have locally-defined definitions.

2.1. Data Model Overview

2.1.1. Features

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

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

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
```

Comments:

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

2.1.3. Groupings

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

```
Groupings:
  +-- 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) {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 as to if 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 2.1.3.6 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) {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 as to if 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 2.1.3.3 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 is defines a truststore instance as being composed of certificates and/or public keys, both of which are enabled by "feature" statements. The stucture 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 referenced together.

* For certificates, each certificate is defined by the "trust-anchor-cert-grouping" grouping Section 2.1.3.6 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 2.1.3.3 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 diagram lists all the protocol-accessible nodes defined in the "ietf-truststore" module:

```
module: ietf-truststore
  +--rw truststore
     +--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
     |           +-- 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].

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

* The reason for why "truststore-grouping" exists separate from the protocol-accessible nodes definition is so as 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 secton 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 contents in the example:

```
Certificate Bags
  +-- CA certificates for authenticating a set a remote servers
  +-- EE certificates for authenticating a set a remote servers
  +-- CA certificates for authenticating a set a remote clients
  +-- EE certificates for authenticating a set a 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:

```
<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>

    <!-- CA Certs for Authenticating Servers Using Private PKIs -->
    <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>base64encodedvalue==</cert-data>
      </certificate>
      <certificate>
        <name>Server Cert Issuer #2</name>
        <cert-data>base64encodedvalue==</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>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>My Application #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </certificate-bag>

        <!-- CA 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>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Client Identity Issuer #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </certificate-bag>

        <!-- 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>base64encodedvalue==</cert-data>
          </certificate>
```

```
        <certificate>
          <name>Fred Flintstone</name>
          <cert-data>base64encodedvalue==</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-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>corp-fw2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>base64encodedvalue==</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-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>Application Instance #2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>base64encodedvalue==</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>base64encodedvalue==</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>base64encodedvalue==</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>base64encodedvalue==</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>base64encodedvalue==</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 2.1.3.4 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.

The following non-normative module is defined to illustrate these groupings:

```
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 "2020-07-10" {
      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.";
      }
    }
  }
```

The tree diagram [RFC8340] for this example module follows:

```
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
     |     |           +-- expiration-date    yang:date-and-time
     |     +--:(truststore) {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) {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>base64encodedvalue==</cert>
         </certificate>
         <certificate>
           <name>Client Identity Issuer #2</name>
           <cert>base64encodedvalue==</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>base64encodedvalue==</public-key>
          </public-key>
          <public-key>
            <name>corp-fw2</name>
            <public-key-format>
              ct:ssh-public-key-format
            </public-key-format>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
        </local-definition>
      </public-key>

   </truststore-usage>
```

2.3. YANG Module

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

<CODE BEGINS> file "ietf-truststore@2020-07-10.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  : <http://datatracker.ietf.org/wg/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) 2020 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 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 2020-07-10 {
      description
        "Initial version";
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

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

    feature truststore-supported {
      description
        "The '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
       defined in the Truststore.";
  }

  typedef certificate-ref {
    type leafref {
      path "/ts:truststore/certificate-bags/certificate-bag" +
           "[name = current()/../certificate-bag]/certificate/name";
    }
    description
      "This typedef define a reference to a specific certificate
       in a certificate bag defined in the Truststore.  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 define a reference to a public key bag
       defined in the Truststore.";
  }
```

```
  typedef public-key-ref {
    type leafref {
      path "/ts:truststore/public-key-bags/public-key-bag" +
           "[name = current()/../public-key-bag]/" +
           "public-key/name";
    }
    description
      "This typedef define a reference to a specific public key
       in a public key bag defined in the Truststore.  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.";
    choice local-or-truststore {
      nacm:default-deny-write;
      mandatory true;
      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 "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.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the Truststore.";
      }
    }


    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.";
      choice local-or-truststore {
        nacm:default-deny-write;
        mandatory true;
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "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 "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 exist
               in the Truststore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the Truststore.";
      }
    }

    grouping truststore-grouping {
      description
        "Grouping definition enables use in other contexts.  Where
         used, implementations SHOULD augment new 'case' statements
         into the local-or-truststore 'choice' statements to supply
         leafrefs to the model-specific location.";
      container certificate-bags {
        nacm:default-deny-write;
        if-feature "certificates";
        presence
          "Indicates that certificate bags have been configured.";
        description
          "A collection of certificate bags.";
        list certificate-bag {
          key "name";
          min-elements 1;
          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";
            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;
              }
            }
          }
        }
      }
      container public-key-bags {
        nacm:default-deny-write;
        if-feature "public-keys";
        presence
          "Indicates that public keys have been configured.";
        description
          "A collection of public key bags.";
        list public-key-bag {
          key "name";
          min-elements 1;
          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";
            min-elements 1;
            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 {
      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 system, they are present in <operational>. 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 CA 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>base64encodedvalue==</cert-data>
         </certificate>
       </certificate-bag>

       <certificate-bag or:origin="or:system">
         <name>Built-In Public CA 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>base64encodedvalue==</cert-data>
         </certificate>
         <certificate>
           <name>Public Root CA Cert 2</name>
           <cert-data>base64encodedvalue==</cert-data>
         </certificate>
         <certificate>
           <name>Public Root CA Cert 3</name>
           <cert-data>base64encodedvalue==</cert-data>
         </certificate>
       </certificate-bag>

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

In order for the built-in trust anchors to be referenced by configuration, the referenced certificates MUST first be copied into <running>. The certificates SHOULD be copied into <running> using the same "key" values, so that the server can bind them to the built-in entries.

Built-in certificates 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 the built-in certificates.

Only the referenced certificates need to be copied; that is, the certificates in <running> MAY be a subset of the built-in certificates define in <operational>. No certificates may be added or changed; that is, the certificates in <running> MUST be a subset (which includes the whole of the set) of the built-in certificates define in <operational>.

A server MUST reject attempts to modify any aspect of built-in trust anchors, both the certificates themselves and the bags that contain them. That these certificates are "configured" in <running> is an illusion, as they are strictly a read-only subset of that which must already exist in <operational>.

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 CA 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>base64encodedvalue==</cert-data>
      </certificate>
    </certificate-bag>

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

4. Security Considerations

4.1. Data at Rest

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

Security controls for the API (i.e., data in motion) are discussed in Section 4.2, but controls for the data at rest 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 signed when persisted to non-volatile memory, to prevent unauthorized modifications from being made undetected.

4.2. 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 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.

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 of 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 RPCs, actions, or notifications, and thus the security consideration 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 NETCONF WG of the IETF.
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 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., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[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-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-17, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-17>.

[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-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[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-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-19>.

[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-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-19>.

[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-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

[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.

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, Eric Voit, Juergen Schoenwaelder, Liang Xia, Martin Bjorklund, and Nick Hancock.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net