NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 21, 2020

K. Watsen
Watsen Networks

May 20, 2020

A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-10

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:

- "2020-05-20" --> the publication date of this draft

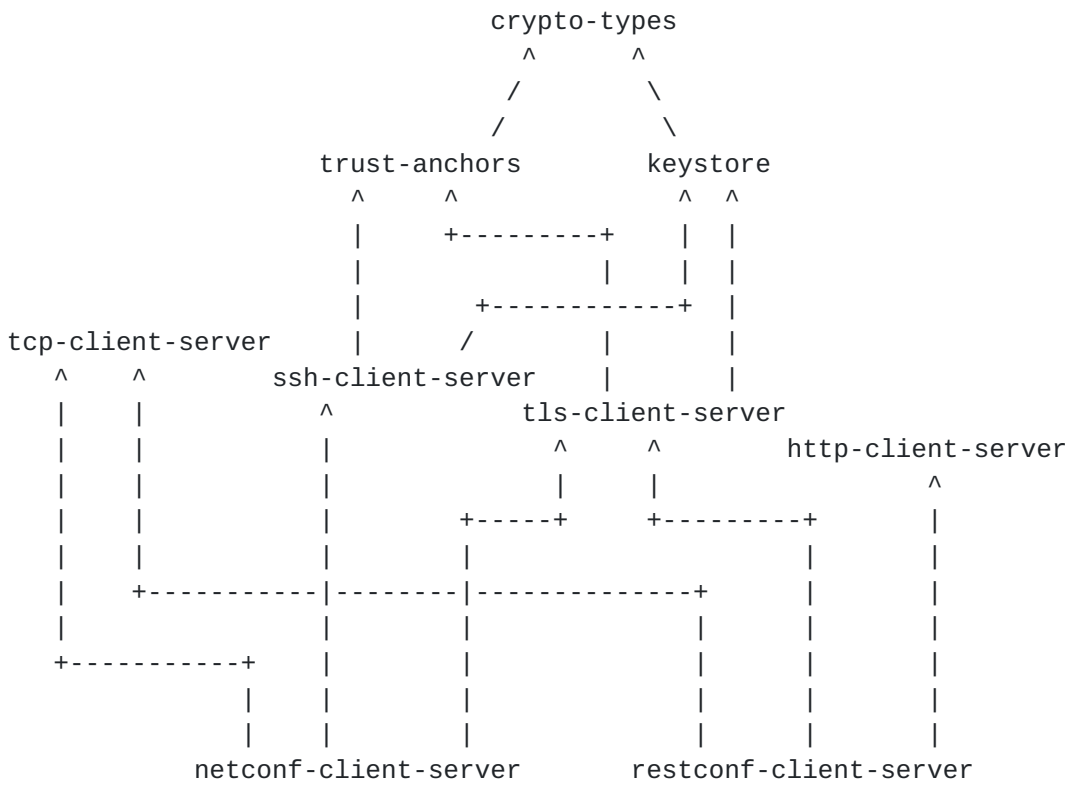
The following Appendix section is to be removed prior to publication:

- Appendix A. Change Log

Note to Reviewers (To be removed by RFC Editor)

This document presents a YANG module or modules that is/are part of a collection of drafts that work together to produce the ultimate goal of the NETCONF WG: to define configuration modules for NETCONF client and servers, and RESTCONF client and servers.

The relationship between the various drafts in the collection is presented in the below diagram.



Full draft names and link to drafts:

- draft-ietf-netconf-crypto-types (html [1])
- draft-ietf-netconf-trust-anchors (html [2])
- draft-ietf-netconf-keystore (html [3])
- draft-ietf-netconf-tcp-client-server (html [4])
- draft-ietf-netconf-ssh-client-server (html [5])
- draft-ietf-netconf-tls-client-server (html [6])
- draft-ietf-netconf-http-client-server (html [7])
- draft-ietf-netconf-netconf-client-server (html [8])
- draft-ietf-netconf-restconf-client-server (html [9])

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 November 21, 2020.

Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 4
1.1. Requirements Language 4
1.2. Tree Diagram Notation 4
2. The Trust Anchors Model 4
2.1. Tree Diagram 4
2.2. Example Usage 6
2.3. YANG Module 11
3. Support for Built-in Trust Anchors 17
4. Security Considerations 20
5. IANA Considerations 21
5.1. The IETF XML Registry 21
5.2. The YANG Module Names Registry 21
6. References 22
6.1. Normative References 22

6.2. Informative References 22
6.3. URIs 23
Appendix A. Change Log 24
A.1. 00 to 01 24
A.2. 01 to 02 24
A.3. 02 to 03 24
A.4. 03 to 04 24
A.5. 04 to 05 24
A.6. 05 to 06 25
A.7. 06 to 07 25
A.8. 07 to 08 25
A.9. 08 to 09 25
A.10. 09 to 10 25
Acknowledgements 25
Author's Address 26

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.

This document in compliant with 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).

1.1. Requirements Language

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

1.2. Tree Diagram Notation

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

2. The Trust Anchors Model

2.1. Tree Diagram

The following tree diagram provides an overview of 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                     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

     grouping local-or-truststore-certs-grouping
       +-- (local-or-truststore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- cert*                     trust-anchor-cert-cms
          |     +---n certificate-expiration
          |        +-- expiration-date    yang:date-and-time
          +--:(truststore) {truststore-supported,certificates}?
             +-- truststore-reference?   ts:certificate-bag-ref
     grouping local-or-truststore-public-keys-grouping
       +-- (local-or-truststore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- public-key* [name]
          |        +-- name?                string
          |        +-- public-key-format    identityref
          |        +-- public-key           binary
          +--:(truststore) {truststore-supported,public-keys}?
             +-- truststore-reference?   ts:public-key-bag-ref
     grouping truststore-grouping
       +-- certificate-bags! {certificates}?
       |  +-- certificate-bag* [name]
       |     +-- name?          string
       |     +-- description?   string
       |     +-- certificate* [name]
       |        +-- name?                     string
       |        +-- cert                      trust-anchor-cert-cms
       |        +---n certificate-expiration
       |           +-- expiration-date    yang:date-and-time
```

```
      +-- public-key-bags! {public-keys}?
         +-- public-key-bag* [name]
            +-- name?          string
            +-- description?   string
            +-- public-key* [name]
               +-- name?                string
               +-- public-key-format    identityref
               +-- public-key           binary
```

2.2. Example Usage

The following example illustrates trust anchors in <intended>. Please see Section 3 for an example illustrating built-in values in <operational>.

========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

```
<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>base64encodedvalue==</cert>
      </certificate>
      <certificate>
        <name>Server Cert Issuer #2</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificate-bag>

    <!-- Pinned 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>base64encodedvalue==</cert>
        </certificate>
        <certificate>
          <name>My Application #2</name>
          <cert>base64encodedvalue==</cert>
        </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>base64encodedvalue==</cert>
        </certificate>
        <certificate>
          <name>Client Identity Issuer #2</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificate-bag>

      <!-- Pinned 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>base64encodedvalue==</cert>
        </certificate>
        <certificate>
          <name>Fred Flintstone</name>
```

```
          <cert>base64encodedvalue==</cert>
        </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>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>corp-fw2</name>
          <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
      </public-key-bag>

      <!-- SSH Public Keys for Authenticating User A -->
      <public-key-bag>
        <name>SSH Public Keys for User A</name>
        <description>
          SSH public keys used to authenticate a user A's SSH public
          keys.  An SSH public key is authenticated if it is an exact
          match to one of these public keys.

          This list of public keys is analogous to OpenSSH's
          "~A/.ssh/authorized_keys" file.
        </description>
        <public-key>
          <name>From Source #1</name>
          <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
```

```
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>From Source #2</name>
          <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
      </public-key-bag>

      <!-- SSH Public Keys for Authenticating User B -->
      <public-key-bag>
        <name>SSH Public Keys for User B</name>
        <description>
          SSH public keys used to authenticate a user B's SSH public
          keys.  An SSH public key is authenticated if it is an exact
          match to one of these public keys.

          This list of public keys is analogous to OpenSSH's
          "~B/.ssh/authorized_keys" file.
        </description>
        <public-key>
          <name>From Source #1</name>
          <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>From Source #2</name>
          <public-key-format>ct:ssh-public-key-format</public-key-form\
at>
          <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>
```

The following example illustrates the "certificate-expiration" notification in use with the NETCONF protocol.

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

<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>explicitly-trusted-client-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>
```

2.3. YANG Module

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

<CODE BEGINS> file "ietf-truststore@2020-05-20.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: Common YANG Data Types 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>
     Author   : Henk Birkholz <henk.birkholz@sit.fraunhofer.de>";

  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 2020-05-20 {
    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 {
        mandatory true;
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "A container for locally configured trust anchor
               certificates.";
            uses ct:trust-anchor-certs-grouping;
          }
        }
        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 {
        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 {
      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" {
                mandatory true;
              }
            }
          }
        }
      }
      container public-key-bags {
        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>base64encodedvalue==</cert>
      </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>base64encodedvalue==</cert>
      </certificate>
      <certificate>
        <name>Public Root CA Cert 2</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
      <certificate>
        <name>Public Root CA Cert 3</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificate-bag>

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

In order for the built-in trust anchors to be referenced by configuration, the referenced nodes MUST first be copied into <running>. They SHOULD be copied into <running> using the same "key" values, so that the system can bind the references to the built-in

entries. Only the referenced nodes need to be copied. When using the same key values as in <operational> no new values can be added and no existing values can be changed; that which is in <running> can only be a subset of that which is in <operational>.

For instance, 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>base64encodedvalue==</cert>
      </certificate>
    </certificate-bag>

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

4. Security Considerations

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.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/: The entire data tree defined by this module is sensitive to write operations. For instance, the addition or removal of any trust anchor may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for the entire data tree.

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments.

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:       ta
reference:    RFC BBBB
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-14 (work in progress), March 2020.

[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

[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>.

6.3. URIs

[1] https://tools.ietf.org/html/draft-ietf-netconf-crypto-types

[2] https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors

[3] https://tools.ietf.org/html/draft-ietf-netconf-keystore

[4] https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server

[5] https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server

[6] https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server

[7] https://tools.ietf.org/html/draft-ietf-netconf-http-client-server

[8] https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server

[9] https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server

Appendix A. Change Log

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.

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 last name): Martin Bjorklund, Nick Hancock, Balazs Kovacs, Eric Voit, and Liang Xia.

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net