Stream: Internet Engineering Task Force (IETF)
RFC: 9641
Category: Standards Track
Published: August 2024
ISSN: 2070-1721
Author: K. Watsen
Watsen Networks

RFC 9641
A YANG Data Model for a Truststore

Abstract

This document presents 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.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Relation to Other RFCs 3
1.2. Specification Language 4
1.3. Adherence to the NMDA 4
1.4. Conventions 4
2. The "ietf-truststore" Module 5
2.1. Data Model Overview 5
2.2. Example Usage 10
2.3. YANG Module 17
3. Support for Built-In Trust Anchors 24
4. Security Considerations 26
4.1. Security of Data at Rest 26
4.2. Unconstrained Public Key Usage 26
4.3. Considerations for the "ietf-truststore" YANG Module 26
5. IANA Considerations 27
5.1. The IETF XML Registry 27
5.2. The YANG Module Names Registry 27
6. References 27
6.1. Normative References 27
6.2. Informative References 28
Acknowledgements 29
Author's Address 29

1. Introduction

This document presents a YANG 1.1 [RFC7950] module that has 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 inline or as references to 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 a YANG module [RFC7950] that is part of a collection of RFCs that work together to ultimately support the configuration of both the clients and servers of both the Network Configuration Protocol (NETCONF) [RFC6241] and RESTCONF [RFC8040].

The dependency relationship between the primary YANG groupings defined in the various RFCs is presented in the below diagram. In some cases, a document may define secondary groupings that introduce dependencies not illustrated in the diagram. The labels in the diagram are shorthand names for the defining RFCs. The citation references for shorthand names are provided below the diagram.

Please note that the arrows in the diagram point from referencer to referenced. For example, the "crypto-types" RFC does not have any dependencies, whilst the "keystore" RFC depends on the "crypto-types" RFC.

```
                               crypto-types
                                 ^      ^
                                /        \
                               /          \
                      truststore         keystore
                       ^     ^             ^  ^
                       |     +---------+   |  |
                       |               |   |  |
                       |      +------------+  |
tcp-client-server      |     /         |      |
   ^    ^        ssh-client-server     |      |
   |    |           ^            tls-client-server
   |    |           |              ^     ^        http-client-server
   |    |           |              |     |                 ^
   |    |           |        +-----+     +---------+       |
   |    |           |        |                     |       |
   |    +-----------|--------|--------------+      |       |
   |                |        |              |      |       |
   +-----------+    |        |              |      |       |
               |    |        |              |      |       |
               |    |        |              |      |       |
            netconf-client-server       restconf-client-server
```

Label in Diagram	Reference

crypto-types	[RFC9640]
truststore	RFC 9641
keystore	[RFC9642]
tcp-client-server	[RFC9643]
ssh-client-server	[RFC9644]
tls-client-server	[RFC9645]
http-client-server	[HTTP-CLIENT-SERVER]
netconf-client-server	[NETCONF-CLIENT-SERVER]
restconf-client-server	[RESTCONF-CLIENT-SERVER]

Table 1: Label in Diagram 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 in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (see Section 4 of [RFC4648]). This placeholder value is used because real base64-encoded structures are often many lines long and hence distracting to the example being presented.

This document uses the adjective "central" with the word "truststore" to refer to the top-level instance of the "truststore-grouping" grouping when the "central-truststore-supported" feature is enabled. Please be aware that consuming YANG modules **MAY** instantiate the "truststore-grouping" grouping in other locations. All such other instances are not the "central" instance.

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 Section 2.2 ("Example Usage"). 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
  +-- inline-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
    +-- central-certificate-bag-ref
    +-- central-certificate-ref
    +-- central-public-key-bag-ref
    +-- central-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 to aid consuming modules that import the "ietf-truststore" module.

2.1.3. Groupings

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

- central-certificate-ref-grouping
- central-public-key-ref-grouping
- inline-or-truststore-certs-grouping
- inline-or-truststore-public-keys-grouping
- truststore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "central-certificate-ref-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "central-certificate-ref-grouping" grouping:

```
  grouping central-certificate-ref-grouping:
    +-- certificate-bag?   ts:central-certificate-bag-ref
    |       {central-truststore-supported,certificates}?
    +-- certificate?       ts:central-certificate-ref
            {central-truststore-supported,certificates}?
```

Comments:

- The "central-certificate-ref-grouping" grouping is provided solely as a convenience to consuming modules that wish to enable the configuration of a reference to a certificate in a certificate-bag in the truststore.
- The "certificate-bag" leaf uses the "central-certificate-bag-ref" typedef defined in Section 2.1.2.
- The "certificate" leaf uses the "central-certificate-ref" typedef defined in Section 2.1.2.

2.1.3.2. The "central-public-key-ref-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "central-public-key-ref-grouping" grouping:

```
  grouping central-public-key-ref-grouping:
    +-- public-key-bag?   ts:central-public-key-bag-ref
    |       {central-truststore-supported,public-keys}?
    +-- public-key?       ts:central-public-key-ref
            {central-truststore-supported,public-keys}?
```

Comments:

- The "central-public-key-ref-grouping" grouping is provided solely as a convenience to consuming modules that wish to enable the configuration of a reference to a public-key in a public-key-bag in the truststore.

- The "public-key-bag" leaf uses the "public-key-bag-ref" typedef defined in Section 2.1.2.
- The "public-key" leaf uses the "public-key-ref" typedef defined in Section 2.1.2.

2.1.3.3. The "inline-or-truststore-certs-grouping" Grouping

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

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

Comments:

- The "inline-or-truststore-certs-grouping" grouping is provided solely as a convenience to consuming modules that wish to offer an option whether a bag of certificates can be defined inline 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 "inline-definition" option, the "certificate" node uses the "trust-anchor-cert-grouping" grouping discussed in Section 2.1.4.8 of [RFC9640].
- For the "central-truststore" option, the "central-truststore-reference" is an instance of the "certificate-bag-ref" discussed in Section 2.1.2.

2.1.3.4. The "inline-or-truststore-public-keys-grouping" Grouping

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

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

Comments:

- The "inline-or-truststore-public-keys-grouping" grouping is provided solely as a convenience to consuming modules that wish to offer an option whether a bag of public keys can be defined inline 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 "inline-definition" option, the "public-key" node uses the "public-key-grouping" grouping discussed in Section 2.1.4.4 of [RFC9640].
- For the "central-truststore" option, the "central-truststore-reference" is an instance of the "certificate-bag-ref" discussed in Section 2.1.2.

2.1.3.5. 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 structures supporting certificates and public keys are essentially the same, having an outer list of "bags" containing an inner list of objects (i.e., 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 2.1.4.8 of [RFC9640]). The "cert-data" node is a Cryptographic Message Syntax (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 expiration or have already expired.
- For public keys, each public key is defined by the "public-key-grouping" grouping (Section 2.1.4.4 of [RFC9640]). 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 protocol-accessible nodes for the "ietf-truststore" module are instances of the "truststore-grouping" grouping discussed in Section 2.1.3.5.
- The top-level "truststore" node is additionally constrained by the "central-truststore-supported" feature.
- The "truststore-grouping" grouping is discussed in Section 2.1.3.5.
- The reason for why the "truststore-grouping" 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 servers
  +-- SSH keys to authenticate a set of remote SSH clients
  +-- Raw public keys to authenticate a set of remote SSH servers
  +-- 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 2.1.4.7 of [RFC9640]) for a certificate configured in the truststore described 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>2024-01-05T14:18:53-05:00</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 "inline-or-truststore" groupings defined in the "ietf-truststore" module, specifically the "inline-or-truststore-certs-grouping" (Section 2.1.3.3) and "inline-or-truststore-public-keys-grouping" (Section 2.1.3.4) groupings.

These examples assume the existence of an example module called "ex-truststore-usage" that has the namespace "https://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 "inline-or-truststore" groupings in use, followed by the YANG module itself.

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

```
module: ex-truststore-usage
  +--rw truststore-usage
     +--rw cert* [name]
     |  +--rw name                                     string
     |  +---u ts:inline-or-truststore-certs-grouping
     +--rw public-key* [name]
        +--rw name                                           string
        +---u ts:inline-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 (inline-or-truststore)
     |     +--:(inline) {inline-definitions-supported}?
     |     |  +--rw inline-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
     |     +--:(central-truststore)
     |              {central-truststore-supported,certificates}?
     |        +--rw central-truststore-reference?
     |                ts:central-certificate-bag-ref
     +--rw public-key* [name]
        +--rw name                                  string
        +--rw (inline-or-truststore)
           +--:(inline) {inline-definitions-supported}?
           |  +--rw inline-definition
           |     +--rw public-key* [name]
           |        +--rw name                 string
           |        +--rw public-key-format    identityref
           |        +--rw public-key           binary
           +--:(central-truststore)
                    {central-truststore-supported,public-keys}?
              +--rw central-truststore-reference?
                      ts:central-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 defined inline. 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 inlined 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="https://example.com/ns/example-truststore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

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

  <cert>
    <name>example 1a</name>
    <central-truststore-reference>trusted-client-ca-certs</central-t\
ruststore-reference>
  </cert>

  <cert>
```

```
    <name>example 1b</name>
    <inline-definition>
      <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>
    </inline-definition>
  </cert>


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

  <public-key>
    <name>example 2a</name>
    <central-truststore-reference>trusted-ssh-public-keys</central-t\
ruststore-reference>
  </public-key>

  <public-key>
    <name>example 2b</name>
    <inline-definition>
      <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>
    </inline-definition>
  </public-key>

</truststore-usage>
```

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

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

  import ietf-truststore {
    prefix ts;
    reference
      "RFC 9641: A YANG Data Model for a Truststore";
  }
  organization
    "Example Corporation";

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

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

  revision 2024-03-16 {
    description
      "Initial version.";
    reference
      "RFC 9641: 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:inline-or-truststore-certs-grouping;
      description
        "A 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:inline-or-truststore-public-keys-grouping;
      description
        "A 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 [RFC9640].

```
<CODE BEGINS> file "ietf-truststore@2024-03-16.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 9640: 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.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
     'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
     'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
     are to be interpreted as described in BCP 14 (RFC 2119)
     (RFC 8174) when, and only when, they appear in all
     capitals, as shown here.

     Copyright (c) 2024 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 9641
     (https://www.rfc-editor.org/info/rfc9641); see the RFC
     itself for full legal notices.";

  revision 2024-03-16 {
    description
      "Initial version.";
    reference
      "RFC 9641: 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 inline-definitions-supported {
      description
        "The 'inline-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 central-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 central truststore.";
    }

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

    typedef central-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 central truststore.";
    }

    typedef central-public-key-ref {
      type leafref {
        path "/ts:truststore/ts:public-key-bags/ts:public-key-bag"
           + "[ts:name = current()/../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.  This typedef
         requires that there exist a sibling 'leaf' node called
         'public-key-bag' SHOULD have the
         'central-public-key-bag-ref' typedef.";
    }

    /*****************/
    /*   Groupings   */
    /*****************/
    // *-ref groupings

    grouping central-certificate-ref-grouping {
      description
        "Grouping for the reference to a certificate in a
         certificate-bag in the central truststore.";
      leaf certificate-bag {
        nacm:default-deny-write;
        if-feature "central-truststore-supported";
        if-feature "certificates";
        type ts:central-certificate-bag-ref;
        must '../certificate';
        description
          "Reference to a certificate-bag in the truststore.";
      }
      leaf certificate {
        nacm:default-deny-write;
        if-feature "central-truststore-supported";
        if-feature "certificates";
        type ts:central-certificate-ref;
        must '../certificate-bag';
        description
          "Reference to a specific certificate in the
           referenced certificate-bag.";
      }
    }

    grouping central-public-key-ref-grouping {
      description
        "Grouping for the reference to a public key in a
         public-key-bag in the central truststore.";
      leaf public-key-bag {
        nacm:default-deny-write;
        if-feature "central-truststore-supported";
        if-feature "public-keys";
        type ts:central-public-key-bag-ref;
        description
          "Reference of a public key bag in the truststore, including
           the certificate to authenticate the TLS client.";
      }
      leaf public-key {
        nacm:default-deny-write;
        if-feature "central-truststore-supported";
        if-feature "public-keys";
        type ts:central-public-key-ref;
        description
          "Reference to a specific public key in the
           referenced public-key-bag.";
      }
    }

    // inline-or-truststore-* groupings

    grouping inline-or-truststore-certs-grouping {
      description
        "A grouping for the configuration of a list of certificates.
         The list of certificates may be defined inline or as a
         reference to a certificate bag in the central truststore.

         Servers that wish to define alternate truststore locations
         MUST augment in custom 'case' statements, enabling
         references to those alternate truststore locations.";
      choice inline-or-truststore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the truststore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "A container for locally configured trust anchor
               certificates.";
            list certificate {
              key "name";
              min-elements 1;
              description
                "A trust anchor certificate or chain of certificates.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this certificate.";
              }
              uses ct:trust-anchor-cert-grouping {
                refine "cert-data" {
                  mandatory true;
                }
              }
            }
          }
        }
        case central-truststore {
          if-feature "central-truststore-supported";
          if-feature "certificates";
          leaf central-truststore-reference {
            type ts:central-certificate-bag-ref;
            description
              "A reference to a certificate bag that exists in the
               central truststore.";
          }
        }
      }
    }

    grouping inline-or-truststore-public-keys-grouping {
      description
        "A grouping that allows the public keys to either be
         configured locally, within the data model being used, or be a
         reference to a public key bag stored in the truststore.

         Servers that wish to define alternate truststore locations
         SHOULD augment in custom 'case' statement, enabling
         references to those alternate truststore locations.";
      choice inline-or-truststore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the truststore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-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 central-truststore {
          if-feature "central-truststore-supported";
          if-feature "public-keys";
          leaf central-truststore-reference {
            type ts:central-public-key-bag-ref;
            description
              "A reference to a bag of public keys that exists
               in the central truststore.";
          }
        }
      }
    }

    // the truststore grouping

    grouping truststore-grouping {
      description
```

```
        "A grouping definition that enables use in other contexts.
         Where used, implementations MUST augment new 'case'
         statements into the various inline-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 or chain of certificates.";
            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
             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 public keys.";
          }
          leaf description {
            type string;
            description
              "A description for this bag of public keys.  The
               intended purpose for the bag MUST 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., a Secure Zero Touch Provisioning (SZTP) [RFC8572] bootstrap server) or public certificate authority (CA) certificates to connect to arbitrary web services using public PKI.

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

The primary characteristic of the built-in trust anchors is that they are provided by the server, as opposed to configuration. As such, they are present in <operational> (Section 5.3 of [RFC8342]) and <system> [NETMOD-SYSTEM-CONFIG], if implemented.

The example below illustrates what the truststore in <operational> might look like for a server in its factory default state. Note that the built-in trust anchor bags have the "or:origin" annotation value "or:system".

```
<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>
```

4. Security Considerations

4.1. Security of Data at Rest

The YANG module specified 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", server implementations **MUST** 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, or both).

Trust anchors configured via this module are implicitly trusted to validate certification paths that may include any name, be used for any purpose, or be 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 a trust anchor.

4.3. Considerations for the "ietf-truststore" YANG Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

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., Secure Shell (SSH), TLS) with mutual authentication.

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

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the security considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

Most of the readable data nodes defined in this YANG module are not considered sensitive or vulnerable in network environments. However, the "cert-data" node uses the NACM "default-deny-all" extension for reasons described in Section 3.8 of [RFC9640].

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 "default-deny-write" extension 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 are not provided here.

5. IANA Considerations

5.1. The IETF XML Registry

IANA has registered the following URI in the "ns" registry defined of the "IETF XML Registry" [RFC3688].

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

IANA has registered the following YANG module in the "YANG Module Names" registry defined in [RFC6020].

Name: ietf-truststore
Namespace: urn:ietf:params:xml:ns:yang:ietf-truststore
Prefix: ts
Reference: RFC 9641

6. References

6.1. Normative References

[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>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

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

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

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

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

[RFC9640] Watsen, K., "YANG Data Types and Groupings for Cryptography", RFC 9640, DOI 10.17487/RFC9640, August 2024, <https://www.rfc-editor.org/info/rfc9640>.

6.2. Informative References

[HTTP-CLIENT-SERVER] Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-23, 15 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-23>.

[NETCONF-CLIENT-SERVER] Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-37, 14 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-37>.

[NETMOD-SYSTEM-CONFIG] Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-08, 18 June 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-08>.

[RESTCONF-CLIENT-SERVER] Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-38, 14 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-38>.

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[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>.

[RFC9642] Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", RFC 9642, DOI 10.17487/RFC9642, August 2024, <https://www.rfc-editor.org/info/rfc9642>.

[RFC9643] Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", RFC 9643, DOI 10.17487/RFC9643, August 2024, <https://www.rfc-editor.org/info/rfc9643>.

[RFC9644] Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", RFC 9644, DOI 10.17487/RFC9644, August 2024, <https://www.rfc-editor.org/info/rfc9644>.

[RFC9645] Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", RFC 9645, DOI 10.17487/RFC9645, August 2024, <https://www.rfc-editor.org/info/rfc9645>.

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 through 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): Balázs Kovács, Carl Wallace, Eric Voit, Éric Vyncke, Francesca Palombini, Jensen Zhang, Jürgen Schönwälder, Lars Eggert, Liang Xia, Martin Björklund, Murray Kucherawy, Nick Hancock, Paul Kyzivat, Qin Wu, Rob Wilton, Robert Varga, Roman Danyliw, and Yoav Nir.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net