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

K. Watsen
Watsen Networks

10 July 2020

YANG Data Types and Groupings for Cryptography
draft-ietf-netconf-crypto-types-17

Abstract

This document presents a YANG 1.1 (RFC 7950) module defining identities, typedefs, and groupings useful to cryptographic applications.

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 this draft

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

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

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

* Appendix A. Change Log

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

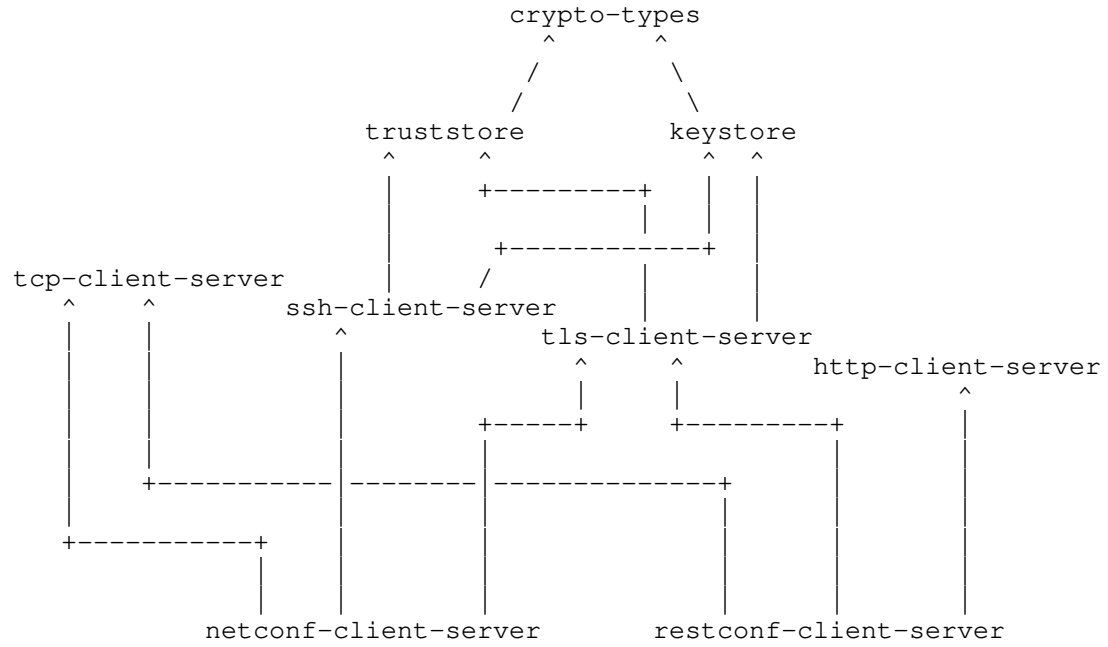
This document presents a YANG 1.1 [RFC7950] module defining identities, typedefs, and groupings useful to cryptographic applications.

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

2. The "ietf-crypto-types" Module

This section defines a YANG 1.1 [RFC7950] module that defines data types (typedefs and identities) and groupings supporting downstream models needing cryptographic primitives.

2.1. Data Model Overview

2.1.1. Features

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

```
Features:
  +-- one-symmetric-key-format
  +-- one-asymmetric-key-format
  +-- encrypted-one-symmetric-key-format
  +-- encrypted-one-asymmetric-key-format
  +-- certificate-signing-request-generation
```

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-crypto-types" module:

```
Identities:
  +-- public-key-format
  |  +-- subject-public-key-info-format
  |  +-- ssh-public-key-format
  +-- private-key-format
  |  +-- rsa-private-key-format
  |  +-- ec-private-key-format
  |  +-- one-asymmetric-key-format
  |  |          {one-asymmetric-key-format}?
  |  +-- encrypted-one-asymmetric-key-format
  |             {encrypted-one-asymmetric-key-format}?
  +-- symmetric-key-format
     +-- octet-string-key-format
     +-- one-symmetric-key-format
     |          {one-symmetric-key-format}?
     +-- encrypted-one-symmetric-key-format
                {encrypted-one-symmetric-key-format}?
```

Comments:

* The diagram shows that there are three base identities. These identities are used by this module to define the format that key data is encoded in. The base identities are "abstract", in the object orientied programming sense, in that they only define a "class" of formats, rather than a specific format.

* The various derived identities define specific key encoding formats. The derived identities defined in this document are sufficient for the effort described in Section 1.1 but, by nature of them being identities, additional derived identities MAY be defined by future efforts.

* Identities use to specify uncommon formats are enabled by "feature" statements, enabling applications to support them when needed.

2.1.3. Typedefs

The following diagram illustrates the relationship amongst the "typedef" statements defined in the "ietf-crypto-types" module:

```
Typedefs:
  binary
    +-- csr-info
    +-- csr
    +-- x509
    |  +-- trust-anchor-cert-x509
    |  +-- end-entity-cert-x509
    +-- crl
    +-- ocsp-request
    +-- ocsp-response
    +-- cms
       +-- data-content-cms
       +-- signed-data-cms
       |  +-- trust-anchor-cert-cms
       |  +-- end-entity-cert-cms
       +-- enveloped-data-cms
       +-- digested-data-cms
       +-- encrypted-data-cms
       +-- authenticated-data-cms
```

Comments:

* All of the typedefs defined in the "ietf-crypto-types" module extend the "binary" type defined in [RFC7950].

* Additionally, all the typedefs define a type for encoding an ASN.1 [ITU.X680.2015] structure using DER [ITU.X690.2015].

* The "trust-anchor-*" and "end-entity-*" typedefs are syntactically identical to their base typedefs and only distiguish themselves by the expected nature of their content. These typedefs are defined to facilitate common modeling needs.

2.1.4. Groupings

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

```
Groupings:
  +-- encrypted-key-value-grouping
  +-- symmetric-key-grouping
  +-- public-key-grouping
  +-- asymmetric-key-pair-grouping
  +-- trust-anchor-cert-grouping
  +-- end-entity-cert-grouping
  +-- generate-csr-grouping
  +-- asymmetric-key-pair-with-cert-grouping
  +-- asymmetric-key-pair-with-certs-grouping
```

Each of these groupings are presented in the following subsections.

2.1.4.1. The "encrypted-key-value-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "encrypted-key-value-grouping" grouping:

```
  grouping encrypted-key-value-grouping
    +-- encrypted-by
    +-- encrypted-value    binary
```

Comments:

* The "encrypted-by" node is an empty container (difficult to see in the diagram) that a consuming module MUST augment key references into. The "ietf-crypto-types" module is unable to populate this container as the module only defines groupings. Section 2.2.1 presents an example illustrating a consuming module populating the "encrypted-by" container.

* The "encrypted-value" node is the key, encrypted by the other key referenced by the "encrypted-by" node, encoded in the format specified by the "format" identity Section 2.1.2 associated with the ancestor node using this grouping.

2.1.4.2. The "symmetric-key-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "symmetric-key-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping symmetric-key-grouping
    +-- key-format?            identityref
    +-- (key-type)
       +--:(cleartext-key)
       |  +-- cleartext-key?   binary
       +--:(hidden-key)
       |  +-- hidden-key?      empty
       +--:(encrypted-key)
          +-- encrypted-key
             +---u encrypted-key-value-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping symmetric-key-grouping
    +-- key-format?            identityref
    +-- (key-type)
       +--:(cleartext-key)
       |  +-- cleartext-key?   binary
       +--:(hidden-key)
       |  +-- hidden-key?      empty
       +--:(encrypted-key)
          +-- encrypted-key
             +-- encrypted-by
             +-- encrypted-value    binary
```

Comments:

* For the referenced grouping statement(s):

  - The "encrypted-key-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "key-format" node is an identity-reference to the "symmetric-key-format" abstract base identity discussed in Section 2.1.2, enabling the symmetric key to be encoded using the format defined by any of the derived identities.

* The "choice" statement enables the private key data to be plain-text, encrypted, or hidden, as follows:

  - The "key" node can encode any plain-text key value.
  - The "hidden-key" node is of type "empty" as the real value cannot be presented via the management interface.
  - The "encrypted-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.3. The "public-key-grouping" Grouping

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

```
  grouping public-key-grouping
    +-- public-key-format    identityref
    +-- public-key           binary
```

Comments:

* The "public-key-format" node is an identity-reference to the "public-key-format" abstract base identity discussed in Section 2.1.2, enabling the public key to be encoded using the format defined by any of the derived identities.

* The "public-key" node is the public key data in the selected format. No "choice" statement is used to hide or encrypt the public key data because it is unecessary to do so for public keys.

2.1.4.4. The "asymmetric-key-pair-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-grouping
    +---u public-key-grouping
    +-- private-key-format?            identityref
    +-- (private-key-type)
       +--:(cleartext-private-key)
       |  +-- cleartext-private-key?   binary
       +--:(hidden-private-key)
       |  +-- hidden-private-key?      empty
       +--:(encrypted-private-key)
          +-- encrypted-private-key
             +---u encrypted-key-value-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping asymmetric-key-pair-grouping
    +-- public-key-format              identityref
    +-- public-key                     binary
    +-- private-key-format?            identityref
    +-- (private-key-type)
       +--:(cleartext-private-key)
       |  +-- cleartext-private-key?   binary
       +--:(hidden-private-key)
       |  +-- hidden-private-key?      empty
       +--:(encrypted-private-key)
          +-- encrypted-private-key
             +-- encrypted-by
             +-- encrypted-value    binary
```

Comments:

* For the referenced grouping statement(s):

  - The "public-key-grouping" grouping is discussed in Section 2.1.4.3.
  - The "encrypted-key-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "private-key-format" node is an identity-reference to the "private-key-format" abstract base identity discussed in Section 2.1.2, enabling the private key to be encoded using the format defined by any of the derived identities.

* The "choice" statement enables the private key data to be plain-text, encrypted, or hidden, as follows:

  - The "private-key" node can encode any plain-text key value.
  - The "hidden-private-key" node is of type "empty" as the real value cannot be presented via the management interface.
  - The "encrypted-private-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.5. The "certificate-expiration-grouping" Grouping

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

```
  grouping certificate-expiration-grouping
    +---n certificate-expiration
       +-- expiration-date    yang:date-and-time
```

Comments:

* This grouping's only purpose is to define the "certificate-expiration" notification statement, used by the groupings defined in Section 2.1.4.6 and Section 2.1.4.7.

* The "certificate-expiration" notification enables servers to notify clients when certificates are nearing expiration.

* The "expiration-date" node indicates when the designated certificate will (or did) expire.

* Identification of the certificate that is expiring is built into the notification itself. For an example, please see Section 2.2.3.

2.1.4.6. The "trust-anchor-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "trust-anchor-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping trust-anchor-cert-grouping
    +-- cert-data?                          trust-anchor-cert-cms
    +---u certificate-expiration-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping trust-anchor-cert-grouping
    +-- cert-data?               trust-anchor-cert-cms
    +---n certificate-expiration
       +-- expiration-date    yang:date-and-time
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.5.

* The "cert-data" node contains a chain of one or more certificates encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.7. The "end-entity-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "end-entity-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping end-entity-cert-grouping
    +-- cert-data?                          end-entity-cert-cms
    +---u certificate-expiration-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping end-entity-cert-grouping
    +-- cert-data?               end-entity-cert-cms
    +---n certificate-expiration
       +-- expiration-date    yang:date-and-time
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.5.

* The "cert-data" node contains a chain of one or more certificates encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.8. The "generate-csr-grouping" Grouping

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

```
  grouping generate-csr-grouping
    +---x generate-certificate-signing-request
            {certificate-signing-request-generation}?
       +---w input
       |  +---w csr-info    ct:csr-info
       +--ro output
          +--ro certificate-signing-request    ct:csr
```

Comments:

* This grouping's only purpose is to define the "generate-certificate-signing-request" action statement, used by the groupings defined in Section 2.1.4.9 and Section 2.1.4.10.

* This action takes as input a "csr-info" type and returns a "csr" type, both of which are discussed in Section 2.1.3.

* For an example, please see Section 2.2.2.

2.1.4.9. The "asymmetric-key-pair-with-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-with-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-with-cert-grouping
    +---u asymmetric-key-pair-grouping
    +---u end-entity-cert-grouping
    +---u generate-csr-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping asymmetric-key-pair-with-cert-grouping
    +-- public-key-format                        identityref
    +-- public-key                               binary
    +-- private-key-format?                      identityref
    +-- (private-key-type)
    |  +--:(cleartext-private-key)
    |  |  +-- cleartext-private-key?             binary
    |  +--:(hidden-private-key)
    |  |  +-- hidden-private-key?                empty
    |  +--:(encrypted-private-key)
    |     +-- encrypted-private-key
    |        +-- encrypted-by
    |        +-- encrypted-value    binary
    +-- cert-data?                               end-entity-cert-cms
    +---n certificate-expiration
    |  +-- expiration-date    yang:date-and-time
    +---x generate-certificate-signing-request
            {certificate-signing-request-generation}?
       +---w input
       |  +---w csr-info    ct:csr-info
       +--ro output
          +--ro certificate-signing-request    ct:csr
```

Comments:

* This grouping defines an asymmetric key with at most one associated certificate, a commonly needed combination in protocol models.

* For the referenced grouping statement(s):

  - The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.4.
  - The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.7.
  - The "generate-csr-grouping" grouping is discussed in Section 2.1.4.8.

2.1.4.10. The "asymmetric-key-pair-with-certs-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-with-certs-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-with-certs-grouping
    +---u asymmetric-key-pair-grouping
    +-- certificates
    |  +-- certificate* [name]
    |     +-- name?                        string
    |     +---u end-entity-cert-grouping
    +---u generate-csr-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping asymmetric-key-pair-with-certs-grouping
    +-- public-key-format                        identityref
    +-- public-key                               binary
    +-- private-key-format?                      identityref
    +-- (private-key-type)
    |  +--:(cleartext-private-key)
    |  |  +-- cleartext-private-key?             binary
    |  +--:(hidden-private-key)
    |  |  +-- hidden-private-key?                empty
    |  +--:(encrypted-private-key)
    |     +-- encrypted-private-key
    |        +-- encrypted-by
    |        +-- encrypted-value    binary
    +-- certificates
    |  +-- certificate* [name]
    |     +-- name?                       string
    |     +-- cert-data                   end-entity-cert-cms
    |     +---n certificate-expiration
    |        +-- expiration-date    yang:date-and-time
    +---x generate-certificate-signing-request
            {certificate-signing-request-generation}?
       +---w input
       |  +---w csr-info    ct:csr-info
       +--ro output
          +--ro certificate-signing-request    ct:csr
```

Comments:

* This grouping defines an asymmetric key with one or more associated certificates, a commonly needed combination in configuration models.

* For the referenced grouping statement(s):

  - The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.4.

- The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.7.
- The "generate-csr-grouping" grouping is discussed in Section 2.1.4.8.

2.1.5. Protocol-accessible Nodes

The "ietf-crypto-types" module does not contain any protocol-accessible nodes, but the module needs to be "implemented", as described in Section 5.6.5 of [RFC7950], in order for the identities in Section 2.1.2 to be defined.

2.2. Example Usage

2.2.1. The "symmetric-key-grouping" and "asymmetric-key-pair-with-certs-grouping" Grouping

The following non-normative module is constructed in order to illustrate the use of the "symmetric-key-grouping" (Section 2.1.4.2) and the "asymmetric-key-pair-with-certs-grouping" (Section 2.1.4.10) grouping statements:

```
module ex-crypto-types-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-crypto-types-usage";
  prefix "ectu";

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

  organization "Example Corporation";
  contact      "YANG Designer <mailto:yang.designer@example.com>";

  description
   "This module illustrates the 'symmetric-key-grouping'
    and 'asymmetric-key-grouping' groupings defined in
    the 'ietf-crypto-types' module defined in RFC AAAA.";

  revision "2020-07-10" {
    description
     "Initial version";
    reference
     "RFC AAAA: Common YANG Data Types for Cryptography";
  }
```

```
    container symmetric-keys {
      description
        "A container of symmetric keys.";
      list symmetric-key {
        key name;
        description
          "A symmetric key";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ct:symmetric-key-grouping {
          augment "key-type/encrypted-key/encrypted-key/"
                  + "encrypted-by" {
            description
              "Augments in a choice statement enabling the
               encrypting key to be any other symmetric or
               asymmetric key.";
            uses encrypted-by-choice-grouping;
          }
        }
      }
    }

    container asymmetric-keys {
      description
        "A container of asymmetric keys.";
      list asymmetric-key {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ct:asymmetric-key-pair-with-certs-grouping {
          augment "private-key-type/encrypted-private-key/"
                  + "encrypted-private-key/encrypted-by" {
            description
              "Augments in a choice statement enabling the
               encrypting key to be any other symmetric or
               asymmetric key.";
            uses encrypted-by-choice-grouping;
          }
        }
        description
          "An asymmetric key pair with associated certificates.";
      }
```

```
    }

    grouping encrypted-by-choice-grouping {
      description
        "A grouping that defines a choice enabling references
         to other keys.";
      choice encrypted-by-choice {
        mandatory true;
        description
          "A choice amongst other symmetric or asymmetric keys.";
        case symmetric-key-ref {
          leaf symmetric-key-ref {
            type leafref {
              path "/ectu:symmetric-keys/ectu:symmetric-key/"
                   + "ectu:name";
            }
            description
             "Identifies the symmetric key used to encrypt this key.";
          }
        }
        case asymmetric-key-ref {
          leaf asymmetric-key-ref {
            type leafref {
              path "/ectu:asymmetric-keys/ectu:asymmetric-key/"
                   + "ectu:name";
            }
            description
             "Identifies the asymmetric key used to encrypt this key.";
          }
        }
      }
    }
  }
```

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

```
module: ex-crypto-types-usage
  +--rw symmetric-keys
  |  +--rw symmetric-key* [name]
  |     +--rw name                    string
  |     +--rw key-format?             identityref
  |     +--rw (key-type)
  |        +--:(cleartext-key)
  |        |  +--rw cleartext-key?    binary
  |        +--:(hidden-key)
  |        |  +--rw hidden-key?       empty
  |        +--:(encrypted-key)
  |           +--rw encrypted-key
```

```
      |              +--rw encrypted-by
      |              |  +--rw (encrypted-by-choice)
      |              |     +--:(symmetric-key-ref)
      |              |     |  +--rw symmetric-key-ref?    leafref
      |              |     +--:(asymmetric-key-ref)
      |              |        +--rw asymmetric-key-ref?   leafref
      |              +--rw encrypted-value    binary
      +--rw asymmetric-keys
         +--rw asymmetric-key* [name]
            +--rw name                                    string
            +--rw public-key-format                       identityref
            +--rw public-key                              binary
            +--rw private-key-format?                     identityref
            +--rw (private-key-type)
            |  +--:(cleartext-private-key)
            |  |  +--rw cleartext-private-key?           binary
            |  +--:(hidden-private-key)
            |  |  +--rw hidden-private-key?              empty
            |  +--:(encrypted-private-key)
            |     +--rw encrypted-private-key
            |        +--rw encrypted-by
            |        |  +--rw (encrypted-by-choice)
            |        |     +--:(symmetric-key-ref)
            |        |     |  +--rw symmetric-key-ref?    leafref
            |        |     +--:(asymmetric-key-ref)
            |        |        +--rw asymmetric-key-ref?   leafref
            |        +--rw encrypted-value    binary
            +--rw certificates
            |  +--rw certificate* [name]
            |     +--rw name                       string
            |     +--rw cert-data                  end-entity-cert-cms
            |     +---n certificate-expiration
            |        +-- expiration-date    yang:date-and-time
            +---x generate-certificate-signing-request
                    {certificate-signing-request-generation}?
               +---w input
               |  +---w csr-info    ct:csr-info
               +--ro output
                  +--ro certificate-signing-request    ct:csr

      grouping encrypted-by-choice-grouping
        +-- (encrypted-by-choice)
           +--:(symmetric-key-ref)
           |  +-- symmetric-key-ref?
           |           -> /symmetric-keys/symmetric-key/name
           +--:(asymmetric-key-ref)
              +-- asymmetric-key-ref?
                      -> /asymmetric-keys/asymmetric-key/name
```

Finally, the following example illustrates various symmetric and asymmetric keys as they might appear in confugration:

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

<symmetric-keys
  xmlns="http://example.com/ns/example-crypto-types-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <symmetric-key>
    <name>ex-hidden-symmetric-key</name>
    <hidden-key/>
  </symmetric-key>
  <symmetric-key>
    <name>ex-octet-string-based-symmetric-key</name>
    <key-format>ct:octet-string-key-format</key-format>
    <cleartext-key>base64encodedvalue==</cleartext-key>
  </symmetric-key>
  <symmetric-key>
    <name>ex-one-symmetric-based-symmetric-key</name>
    <key-format>ct:one-symmetric-key-format</key-format>
    <cleartext-key>base64encodedvalue==</cleartext-key>
  </symmetric-key>
  <symmetric-key>
    <name>ex-encrypted-one-symmetric-based-symmetric-key</name>
    <key-format>ct:encrypted-one-symmetric-key-format</key-format>
    <encrypted-key>
      <encrypted-by>
        <asymmetric-key-ref>ex-hidden-asymmetric-key</asymmetric-key\
-ref>
      </encrypted-by>
      <encrypted-value>base64encodedvalue==</encrypted-value>
    </encrypted-key>
  </symmetric-key>
</symmetric-keys>

<asymmetric-keys
  xmlns="http://example.com/ns/example-crypto-types-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <asymmetric-key>
    <name>ex-hidden-asymmetric-key</name>
    <public-key-format>
      ct:subject-public-key-info-format
    </public-key-format>
    <public-key>base64encodedvalue==</public-key>
    <hidden-private-key/>
    <certificates>
      <certificate>
        <name>ex-hidden-asymmetric-key-cert</name>
```

```
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-subject-public-info-based-asymmetric-key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <private-key-format>
        ct:rsa-private-key-format
      </private-key-format>
      <cleartext-private-key>base64encodedvalue==</cleartext-private-k\
ey>
      <certificates>
        <certificate>
          <name>ex-cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-one-asymmetric-based-symmetric-key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <private-key-format>
        ct:one-asymmetric-key-format
      </private-key-format>
      <cleartext-private-key>base64encodedvalue==</cleartext-private-k\
ey>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-encrypted-one-asymmetric-based-symmetric-key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <private-key-format>
        ct:encrypted-one-asymmetric-key-format
      </private-key-format>
      <encrypted-private-key>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
        </encrypted-by>
```

```
        <encrypted-value>base64encodedvalue==</encrypted-value>
      </encrypted-private-key>
    </asymmetric-key>
  </asymmetric-keys>
```

2.2.2. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action, discussed in Section 2.1.4.8, with the NETCONF protocol.

REQUEST

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <asymmetric-keys
      xmlns="http://example.com/ns/example-crypto-types-usage">
      <asymmetric-key>
        <name>ex-key-sect571r1</name>
        <generate-certificate-signing-request>
          <csr-info>base64encodedvalue==</csr-info>
        </generate-certificate-signing-request>
      </asymmetric-key>
    </asymmetric-keys>
  </action>
</rpc>
```

RESPONSE

```
<rpc-reply message-id="101"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <certificate-signing-request
     xmlns="http://example.com/ns/example-crypto-types-usage">
     base64encodedvalue==
   </certificate-signing-request>
</rpc-reply>
```

2.2.3. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification, discussed in Section 2.1.4.5, with the NETCONF protocol.

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

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <asymmetric-keys xmlns="http://example.com/ns/example-crypto-types\
   -usage">
       <asymmetric-key>
         <name>ex-hidden-asymmetric-key</name>
         <certificates>
           <certificate>
             <name>ex-hidden-asymmetric-key</name>
             <certificate-expiration>
               <expiration-date>2018-08-05T14:18:53-05:00</expiration-d\
   ate>
             </certificate-expiration>
           </certificate>
         </certificates>
       </asymmetric-key>
     </asymmetric-keys>
   </notification>
```

2.3. YANG Module

This module has normative references to [RFC2119], [RFC2986], [RFC3447], [RFC4253], [RFC5280], [RFC5652], [RFC5915], [RFC5958], [RFC6031], [RFC6125], [RFC6991], [RFC8174], [RFC8341], and [ITU.X690.2015].

```
   <CODE BEGINS> file "ietf-crypto-types@2020-07-10.yang"

   module ietf-crypto-types {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-crypto-types";
     prefix ct;

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

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

```
  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module defines common YANG types for cryptographic
     applications.

     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 AAAA
     (https://www.rfc-editor.org/info/rfcAAAA); see the RFC
     itself for full legal notices.

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

  revision 2020-07-10 {
    description
      "Initial version";
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }


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

  feature one-symmetric-key-format {
    description
      "Indicates that the server supports the
```

```
         'one-symmetric-key-format' identity.";
    }

    feature one-asymmetric-key-format {
      description
        "Indicates that the server supports the
         'one-asymmetric-key-format' identity.";
    }

    feature encrypted-one-symmetric-key-format {
      description
        "Indicates that the server supports the
         'encrypted-one-symmetric-key-format' identity.";
    }

    feature encrypted-one-asymmetric-key-format {
      description
        "Indicates that the server supports the
         'encrypted-one-asymmetric-key-format' identity.";
    }

    feature certificate-signing-request-generation {
      description
        "Indicates that the server implements the
         'generate-certificate-signing-request' action.";
    }


    /************************************************/
    /*   Base Identities for Key Format Structures   */
    /************************************************/

    identity symmetric-key-format {
      description "Base key-format identity for symmetric keys.";
    }

    identity public-key-format {
      description "Base key-format identity for public keys.";
    }

    identity private-key-format {
      description "Base key-format identity for private keys.";
    }


    /************************************************/
    /*   Identities for Private Key Format Structures   */
    /************************************************/
```

```
identity rsa-private-key-format {
  base "private-key-format";
  description
    "Indicates that the private key value is encoded
     as an RSAPrivateKey (from RFC 3447).";
  reference
    "RFC 3447: PKCS #1: RSA Cryptography
               Specifications Version 2.2";
}

identity ec-private-key-format {
  base "private-key-format";
  description
    "Indicates that the private key value is encoded
     as an ECPrivateKey (from RFC 5915)";
  reference
    "RFC 5915: Elliptic Curve Private Key Structure";
}

identity one-asymmetric-key-format {
  if-feature "one-asymmetric-key-format";
  base "private-key-format";
  description
    "Indicates that the private key value is a CMS
     OneAsymmetricKey structure, as defined in RFC 5958,
     encoded using ASN.1 distinguished encoding rules
     (DER), as specified in ITU-T X.690.";
  reference
    "RFC 5958: Asymmetric Key Packages
     ITU-T X.690:
       Information technology - ASN.1 encoding rules:
       Specification of Basic Encoding Rules (BER),
       Canonical Encoding Rules (CER) and Distinguished
       Encoding Rules (DER).";
}

identity encrypted-one-asymmetric-key-format {
  if-feature "encrypted-one-asymmetric-key-format";
  base "private-key-format";
  description
    "Indicates that the private key value is a CMS EnvelopedData
     structure, per Section 8 in RFC 5652, containing a
     OneAsymmetricKey structure, as defined in RFC 5958,
     encoded using ASN.1 distinguished encoding rules (DER),
     as specified in ITU-T X.690.";
  reference
    "RFC 5652: Cryptographic Message Syntax (CMS)
     RFC 5958: Asymmetric Key Packages
```

```
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
        }


      /*************************************************/
      /*   Identities for Public Key Format Structures   */
      /*************************************************/

      identity ssh-public-key-format {
        base "public-key-format";
        description
          "Indicates that the public key value is an SSH public key,
           as specified by RFC 4253, Section 6.6, i.e.:

             string    certificate or public key format
                       identifier
             byte[n]   key/certificate data.";
        reference
          "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      }

      identity subject-public-key-info-format {
        base "public-key-format";
        description
          "Indicates that the public key value is a SubjectPublicKeyInfo
           structure, as described in RFC 5280 encoded using ASN.1
           distinguished encoding rules (DER), as specified in
           ITU-T X.690.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER).";
      }


      /*****************************************************/
      /*   Identities for Symmetric Key Format Structures   */
      /*****************************************************/
```

```
  identity octet-string-key-format {
    base "symmetric-key-format";
    description
      "Indicates that the key is encoded as a raw octet string.
       The length of the octet string MUST be appropriate for
       the associated algorithm's block size.";
  }

  identity one-symmetric-key-format {
    if-feature "one-symmetric-key-format";
    base "symmetric-key-format";
    description
      "Indicates that the private key value is a CMS
       OneSymmetricKey structure, as defined in RFC 6031,
       encoded using ASN.1 distinguished encoding rules
       (DER), as specified in ITU-T X.690.";
    reference
      "RFC 6031: Cryptographic Message Syntax (CMS)
                 Symmetric Key Package Content Type
       ITU-T X.690:
         Information technology - ASN.1 encoding rules:
         Specification of Basic Encoding Rules (BER),
         Canonical Encoding Rules (CER) and Distinguished
         Encoding Rules (DER).";
  }

  identity encrypted-one-symmetric-key-format {
    if-feature "encrypted-one-symmetric-key-format";
    base "symmetric-key-format";
    description
      "Indicates that the private key value is a CMS
       EnvelopedData structure, per Section 8 in RFC 5652,
       containing a OneSymmetricKey structure, as defined
       in RFC 6031, encoded using ASN.1 distinguished
       encoding rules (DER), as specified in ITU-T X.690.";
    reference
      "RFC 5652: Cryptographic Message Syntax (CMS)
       RFC 6031: Cryptographic Message Syntax (CMS)
                 Symmetric Key Package Content Type
       ITU-T X.690:
         Information technology - ASN.1 encoding rules:
         Specification of Basic Encoding Rules (BER),
         Canonical Encoding Rules (CER) and Distinguished
         Encoding Rules (DER).";
  }


  /*************************************************/
```

```
     /*   Typedefs for ASN.1 structures from RFC 2986   */
     /*************************************************/

     typedef csr-info {
       type binary;
       description
         "A CertificationRequestInfo structure, as defined in
          RFC 2986, encoded using ASN.1 distinguished encoding
          rules (DER), as specified in ITU-T X.690.";
       reference
         "RFC 2986: PKCS #10: Certification Request Syntax
                    Specification Version 1.7
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
     }

     typedef csr {
       type binary;
       description
         "A CertificationRequest structure, as specified in
          RFC 2986, encoded using ASN.1 distinguished encoding
          rules (DER), as specified in ITU-T X.690.";
       reference
         "RFC 2986:
            PKCS #10: Certification Request Syntax Specification
            Version 1.7
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
     }

     /*************************************************/
     /*   Typedefs for ASN.1 structures from RFC 5280   */
     /*************************************************/

     typedef x509 {
       type binary;
       description
         "A Certificate structure, as specified in RFC 5280,
          encoded using ASN.1 distinguished encoding rules (DER),
          as specified in ITU-T X.690.";
       reference
         "RFC 5280:
```

```
            Internet X.509 Public Key Infrastructure Certificate
            and Certificate Revocation List (CRL) Profile
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
      }

      typedef crl {
        type binary;
        description
          "A CertificateList structure, as specified in RFC 5280,
           encoded using ASN.1 distinguished encoding rules (DER),
           as specified in ITU-T X.690.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER).";
      }


      /*************************************************/
      /*   Typedefs for ASN.1 structures from RFC 6960   */
      /*************************************************/

      typedef oscp-request {
        type binary;
        description
          "A OCSPRequest structure, as specified in RFC 6960,
           encoded using ASN.1 distinguished encoding rules
           (DER), as specified in ITU-T X.690.";
        reference
          "RFC 6960:
             X.509 Internet Public Key Infrastructure Online
             Certificate Status Protocol - OCSP
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER).";
      }
```

```
    typedef oscp-response {
      type binary;
      description
        "A OCSPResponse structure, as specified in RFC 6960,
         encoded using ASN.1 distinguished encoding rules
         (DER), as specified in ITU-T X.690.";
      reference
        "RFC 6960:
           X.509 Internet Public Key Infrastructure Online
           Certificate Status Protocol - OCSP
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }


    /***********************************************/
    /*   Typedefs for ASN.1 structures from 5652   */
    /***********************************************/

    typedef cms {
      type binary;
      description
        "A ContentInfo structure, as specified in RFC 5652,
         encoded using ASN.1 distinguished encoding rules (DER),
         as specified in ITU-T X.690.";
      reference
        "RFC 5652:
           Cryptographic Message Syntax (CMS)
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }

    typedef data-content-cms {
      type cms;
      description
        "A CMS structure whose top-most content type MUST be the
         data content type, as described by Section 4 in RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef signed-data-cms {
```

```
      type cms;
      description
        "A CMS structure whose top-most content type MUST be the
         signed-data content type, as described by Section 5 in
         RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef enveloped-data-cms {
      type cms;
      description
        "A CMS structure whose top-most content type MUST be the
         enveloped-data content type, as described by Section 6
         in RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef digested-data-cms {
      type cms;
      description
        "A CMS structure whose top-most content type MUST be the
         digested-data content type, as described by Section 7
         in RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef encrypted-data-cms {
      type cms;
      description
        "A CMS structure whose top-most content type MUST be the
         encrypted-data content type, as described by Section 8
         in RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef authenticated-data-cms {
      type cms;
      description
        "A CMS structure whose top-most content type MUST be the
         authenticated-data content type, as described by Section 9
         in RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }
```

```
     /*********************************************************/
     /*   Typedefs for ASN.1 structures related to RFC 5280   */
     /*********************************************************/

     typedef trust-anchor-cert-x509 {
       type x509;
       description
         "A Certificate structure that MUST encode a self-signed
          root certificate.";
     }

     typedef end-entity-cert-x509 {
       type x509;
       description
         "A Certificate structure that MUST encode a certificate
          that is neither self-signed nor having Basic constraint
          CA true.";
     }


     /*********************************************************/
     /*   Typedefs for ASN.1 structures related to RFC 5652   */
     /*********************************************************/

     typedef trust-anchor-cert-cms {
       type signed-data-cms;
       description
         "A CMS SignedData structure that MUST contain the chain of
          X.509 certificates needed to authenticate the certificate
          presented by a client or end-entity.

          The CMS MUST contain only a single chain of certificates.
          The client or end-entity certificate MUST only authenticate
          to last intermediate CA certificate listed in the chain.

          In all cases, the chain MUST include a self-signed root
          certificate.  In the case where the root certificate is
          itself the issuer of the client or end-entity certificate,
          only one certificate is present.

          This CMS structure MAY (as applicable where this type is
          used) also contain suitably fresh (as defined by local
          policy) revocation objects with which the device can
          verify the revocation status of the certificates.

          This CMS encodes the degenerate form of the SignedData
          structure that is commonly used to disseminate X.509
          certificates and revocation objects (RFC 5280).";
```

```
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile.";
    }

    typedef end-entity-cert-cms {
      type signed-data-cms;
      description
        "A CMS SignedData structure that MUST contain the end
         entity certificate itself, and MAY contain any number
         of intermediate certificates leading up to a trust
         anchor certificate.  The trust anchor certificate
         MAY be included as well.

         The CMS MUST contain a single end entity certificate.
         The CMS MUST NOT contain any spurious certificates.

         This CMS structure MAY (as applicable where this type is
         used) also contain suitably fresh (as defined by local
         policy) revocation objects with which the device can
         verify the revocation status of the certificates.

         This CMS encodes the degenerate form of the SignedData
         structure that is commonly used to disseminate X.509
         certificates and revocation objects (RFC 5280).";
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile.";
    }


    /***********************************************/
    /*   Groupings for keys and/or certificates   */
    /***********************************************/

    grouping encrypted-key-value-grouping {
      description
        "A reusable grouping for a value that has been encrypted by
         a symmetric or asymmetric key in the Keystore.";
      container encrypted-by {
        nacm:default-deny-write;
        description
          "An empty container enabling references to other keys that
           encrypt these keys to be augmented in.  The referenced key
           MAY be a symmetric or an asymmetric key.";
      }
```

```
      leaf encrypted-value {
        nacm:default-deny-write;
        type binary;
        must "../encrypted-by";
        mandatory true;
        description
          "The key data, encrypted using the referenced symmetric
           or asymmetric key.  The format of the encrypted value
           is identicated by the associated key format identity.";
      }
    }

    grouping symmetric-key-grouping {
      description
        "A symmetric key.";
      leaf key-format {
        nacm:default-deny-write;
        type identityref {
          base symmetric-key-format;
        }
        description "Identifies the symmetric key's format.";
      }
      choice key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-key {
          leaf cleartext-key {
            nacm:default-deny-all;
            type binary;
            must "../key-format";
            description
              "The binary value of the key.  The interpretation of
               the value is defined by the 'key-format' field.";
          }
        }
        case hidden-key {
          leaf hidden-key {
            type empty;
            must "not(../key-format)";
            description
              "A hidden key.  How such keys are created is outside
               the scope of this module.";
          }
        }
        case encrypted-key {
          container encrypted-key {
```

```
            must "../key-format";
            description
              "A container for the encrypted symmetric key value.";
            uses encrypted-key-value-grouping;
          }
        }
      }
    }

    grouping public-key-grouping {
      description
        "A public key.";
      leaf public-key-format {
        nacm:default-deny-write;
        type identityref {
          base public-key-format;
        }
        mandatory true;
        description "Identifies the key's format.";
      }
      leaf public-key {
        nacm:default-deny-write;
        type binary;
        mandatory true;
        description
          "The binary value of the public key.  The interpretation
           of the value is defined by 'public-key-format' field.";
      }
    }

    grouping asymmetric-key-pair-grouping {
      description
        "A private key and its associated public key.";
      uses public-key-grouping;
      leaf private-key-format {
        nacm:default-deny-write;
        type identityref {
          base private-key-format;
        }
        description "Identifies the key's format.";
      }
      choice private-key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-private-key {
          leaf cleartext-private-key {
```

```
            nacm:default-deny-all;
            type binary;
            must "../private-key-format";
            description
              "The value of the binary key  The key's value is
               interpreted by the 'private-key-format' field.";
          }
        }
        case hidden-private-key {
          leaf hidden-private-key {
            type empty;
            must "not(../private-key-format)";
            description
              "A hidden key.  How such keys are created is
               outside the scope of this module.";
          }
        }
        case encrypted-private-key {
          container encrypted-private-key {
            must "../private-key-format";
            description
             "A container for the encrypted asymmetric private
               key value.";
            uses encrypted-key-value-grouping;
          }
        }
      }
    }

    grouping certificate-expiration-grouping {
      description
        "A notification for when a certificate is about to, or
        already has, expired.";
      notification certificate-expiration {
        description
          "A notification indicating that the configured certificate
           is either about to expire or has already expired.  When to
           send notifications is an implementation specific decision,
           but it is RECOMMENDED that a notification be sent once a
           month for 3 months, then once a week for four weeks, and
           then once a day thereafter until the issue is resolved.";
        leaf expiration-date {
          type yang:date-and-time;
          mandatory true;
          description
            "Identifies the expiration date on the certificate.";
        }
      }
```

```
  }

  grouping trust-anchor-cert-grouping {
    description
      "A trust anchor certificate, and a notification for when
       it is about to (or already has) expire.";
    leaf cert-data {
      nacm:default-deny-write;
      type trust-anchor-cert-cms;
      description
        "The binary certificate data for this certificate.";
    }
    uses certificate-expiration-grouping;
  }

  grouping end-entity-cert-grouping {
    description
      "An end entity certificate, and a notification for when
       it is about to (or already has) expire.  Implementations
       SHOULD assert that, where used, the end entity certificate
       contains the expected public key.";
    leaf cert-data {
      nacm:default-deny-write;
      type end-entity-cert-cms;
      description
        "The binary certificate data for this certificate.";
    }
    uses certificate-expiration-grouping;
  }

  grouping generate-csr-grouping {
    description
      "Defines the 'generate-certificate-signing-request' action.";
    action generate-certificate-signing-request {
      if-feature certificate-signing-request-generation;
      nacm:default-deny-all;
      description
        "Generates a certificate signing request structure for
         the associated asymmetric key using the passed subject
         and attribute values.

         This action statement is only available when the
         associated 'public-key-format' node's value is
         'subject-public-key-info-format'.";
      reference
        "RFC 6125:
          Representation and Verification of Domain-Based
          Application Service Identity within Internet Public Key
```

```
            Infrastructure Using X.509 (PKIX) Certificates in the
            Context of Transport Layer Security (TLS)";
          input {
            leaf csr-info {
              type ct:csr-info;
              mandatory true;
              description
                "A CertificationRequestInfo structure, as defined in
                 RFC 2986.

                 Enables the client to provide a fully-populated
                 CertificationRequestInfo structure that the server
                 only needs to sign in order to generate the complete
                 'CertificationRequest' structure to return in the
                 'output'.

                 The 'AlgorithmIdentifier' field contained inside
                 the 'SubjectPublicKeyInfo' field MUST be one known
                 to be supported by the device.";
              reference
                "RFC 2986:
                   PKCS #10: Certification Request Syntax Specification
                 RFC AAAA:
                   YANG Data Types and Groupings for Cryptography";
            }
          }
          output {
            leaf certificate-signing-request {
              type ct:csr;
              mandatory true;
              description
                "A CertificationRequest structure, as defined in
                 RFC 2986.";
              reference
                "RFC 2986:
                   PKCS #10: Certification Request Syntax Specification
                 RFC AAAA:
                   YANG Data Types and Groupings for Cryptography";
            }
          }
        }
      } // generate-csr-grouping

      grouping asymmetric-key-pair-with-cert-grouping {
        description
          "A private/public key pair and an associated certificate.
           Implementations SHOULD assert that certificates contain
           the matching public key.";
```

```
      uses asymmetric-key-pair-grouping;
      uses end-entity-cert-grouping;
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-cert-grouping

    grouping asymmetric-key-pair-with-certs-grouping {
      description
        "A private/public key pair and associated certificates.
         Implementations SHOULD assert that certificates contain
         the matching public key.";
      uses asymmetric-key-pair-grouping;
      container certificates {
        nacm:default-deny-write;
        description
          "Certificates associated with this asymmetric key.
           More than one certificate supports, for instance,
           a TPM-protected asymmetric key that has both IDevID
           and LDevID certificates associated.";
        list certificate {
          key "name";
          description
            "A certificate for this asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the certificate.  If the name
               matches the name of a certificate that exists
               independently in <operational> (i.e., an IDevID),
               then the 'cert' node MUST NOT be configured.";
          }
          uses end-entity-cert-grouping {
            refine cert-data {
              mandatory true;
            }
          }
        }
      }
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-certs-grouping

  }

  <CODE ENDS>
```

3. Security Considerations

3.1. No Support for CRMF

This document uses PKCS #10 [RFC2986] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [RFC4211] was considered, but is was unclear if there was market demand for it. If it is desired to support CRMF in the future, a backwards compatible solution can be defined at that time.

3.2. No Support for Key Generation

Early revisions of this document included "rpc" statements for generating symmetric and asymmetric keys. There statements were removed due to an inability to obtain consensus for how to identify the key-algorithm to use. Thusly, the solution presented in this document only supports keys to be configured via an external client, which does not support Security best practice.

3.3. Strength of Keys Configured

When configuring key values, implementations SHOULD ensure that the strength of the key being configured is not greater than the strength of the underlying secure transport connection over which it is communicated. Implementations SHOULD fail the write-request if ever the strength of the private key is greater then the strength of the underlying transport.

3.4. Deletion of Cleartext Key Values

This module defines storage for cleartext key values that SHOULD be zeroized when deleted, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

The cleartext key nodes are the "key" node defined in the "symmetric-key-grouping" grouping (Section 2.1.4.2) and the "private-key" node defined in the "asymmetric-key-pair-grouping" grouping ("Section 2.1.4.4).

3.5. The "ietf-crypto-types" YANG Module

The YANG module in this document defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

Some of the readable data nodes defined in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

* The "key" node:

  The cleartext "key" node defined in the "symmetric-key-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

* The "private-key" node:

  The cleartext "private-key" node defined in the "asymmetric-key-pair-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied.

All of the writable data nodes defined by all the groupings defined in this module may be considered sensitive or vulnerable in some network environments. For instance, even the modification of a public key or a certificate can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been applied to all the data nodes defined in the module.

Some of the operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

* generate-certificate-signing-request:

This "action" statement SHOULD only be executed by authorized users. For this reason, the NACM extension "default-deny-all" has been applied. Note that NACM uses "default-deny-all" to protect "RPC" and "action" statements; it does not define, e.g., an extension called "default-deny-execute".

For this action, it is RECOMMENDED that implementations assert channel binding [RFC5056], so as to ensure that the application layer that sent the request is the same as the device authenticated when the secure transport layer was established.

4. IANA Considerations

4.1. 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-crypto-types
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

4.2. 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-crypto-types
namespace:    urn:ietf:params:xml:ns:yang:ietf-crypto-types
prefix:       ct
reference:    RFC AAAA
```

5. References

5.1. Normative References

[ITU.X680.2015]
International Telecommunication Union, "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2015, August 2015, <https://www.itu.int/rec/T-REC-X.680/>.

[ITU.X690.2015]
International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic

Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2015, August 2015, <https://www.itu.int/rec/T-REC-X.690/>.

[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>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <https://www.rfc-editor.org/info/rfc3447>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC6031] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Symmetric Key Package Content Type", RFC 6031, DOI 10.17487/RFC6031, December 2010, <https://www.rfc-editor.org/info/rfc6031>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[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>.

5.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

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

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

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

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <https://www.rfc-editor.org/info/rfc5056>.

[RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", RFC 5915, DOI 10.17487/RFC5915, June 2010, <https://www.rfc-editor.org/info/rfc5915>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[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>.

Appendix A. Change Log

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

A.1. I-D to 00

* Removed groupings and notifications.
* Added typedefs for identityrefs.
* Added typedefs for other RFC 5280 structures.
* Added typedefs for other RFC 5652 structures.
* Added convenience typedefs for RFC 4253, RFC 5280, and RFC 5652.

A.2. 00 to 01

* Moved groupings from the draft-ietf-netconf-keystore here.

A.3. 01 to 02

* Removed unwanted "mandatory" and "must" statements.

* Added many new crypto algorithms (thanks Haiguang!)

* Clarified in asymmetric-key-pair-with-certs-grouping, in certificates/certificate/name/description, that if the name MUST NOT match the name of a certificate that exists independently in <operational>, enabling certs installed by the manufacturer (e.g., an IDevID).

A.4. 02 to 03

* renamed base identity 'asymmetric-key-encryption-algorithm' to 'asymmetric-key-algorithm'.

* added new 'asymmetric-key-algorithm' identities for secp192r1, secp224r1, secp256r1, secp384r1, and secp521r1.

* removed 'mac-algorithm' identities for mac-aes-128-ccm, mac-aes-192-ccm, mac-aes-256-ccm, mac-aes-128-gcm, mac-aes-192-gcm, mac-aes-256-gcm, and mac-chacha20-poly1305.

* for all -cbc and -ctr identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-algorithm'.

* for all -ccm and -gcm identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-and-mac-algorithm' and renamed the identity to remove the "enc-" prefix.

* for all the 'signature-algorithm' based identities, renamed from 'rsa-*' to 'rsassa-*'.

* removed all of the "x509v3-" prefixed 'signature-algorithm' based identities.

* added 'key-exchange-algorithm' based identities for 'rsaes-oaep' and 'rsaes-pkcs1-v1_5'.

* renamed typedef 'symmetric-key-encryption-algorithm-ref' to 'symmetric-key-algorithm-ref'.

* renamed typedef 'asymmetric-key-encryption-algorithm-ref' to 'asymmetric-key-algorithm-ref'.

* added typedef 'encryption-and-mac-algorithm-ref'.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.5. 03 to 04

* ran YANG module through formatter.

A.6. 04 to 05

* fixed broken symlink causing reformatted YANG module to not show.

A.7. 05 to 06

* Added NACM annotations.

* Updated Security Considerations section.

* Added 'asymmetric-key-pair-with-cert-grouping' grouping.

* Removed text from 'permanently-hidden' enum regarding such keys not being backed up or restored.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Added an explanation to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements as for why the nodes are not mandatory (e.g., because they may exist only in <operational>.

* Added 'must' expressions to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements ensuring sibling nodes are either all exist or do not all exist.

* Added an explanation to the 'permanently-hidden' that the value cannot be configured directly by clients and servers MUST fail any attempt to do so.

* Added 'trust-anchor-certs-grouping' and 'end-entity-certs-grouping' (the plural form of existing groupings).

* Now states that keys created in <operational> by the *-hidden-key actions are bound to the lifetime of the parent 'config true' node, and that subsequent invocations of either action results in a failure.

A.8. 06 to 07

* Added clarifications that implementations SHOULD assert that configured certificates contain the matching public key.

* Replaced the 'generate-hidden-key' and 'install-hidden-key' actions with special 'crypt-hash' -like input/output values.

A.9. 07 to 08

* Removed the 'generate-key and 'hidden-key' features.

* Added grouping symmetric-key-grouping

* Modified 'asymmetric-key-pair-grouping' to have a 'choice' statement for the keystone module to augment into, as well as replacing the 'union' with leafs (having different NACM settings.

A.10. 08 to 09

* Converting algorithm from identities to enumerations.

A.11. 09 to 10

* All of the below changes are to the algorithm enumerations defined in ietf-crypto-types.

* Add in support for key exchange over x.25519 and x.448 based on RFC 8418.

* Add in SHAKE-128, SHAKE-224, SHAKE-256, SHAKE-384 and SHAKE 512

* Revise/add in enum of signature algorithm for x25519 and x448

* Add in des3-cbc-sha1 for IPSec

* Add in sha1-des3-kd for IPSec

* Add in definit for rc4-hmac and rc4-hmac-exp. These two algorithms have been deprecated in RFC 8429. But some existing draft in i2nsf may still want to use them.

* Add x25519 and x448 curve for asymmetric algorithms

* Add signature algorithms ed25519, ed25519-cts, ed25519ph

* add signature algorithms ed448, ed448ph

* Add in rsa-sha2-256 and rsa-sha2-512 for SSH protocols (rfc8332)

A.12. 10 to 11

* Added a "key-format" identity.

* Added symmetric keys to the example in Section 2.2.

A.13. 11 to 12

* Removed all non-essential (to NC/RC) algorithm types.

* Moved remaining algorithm types each into its own module.

* Added a 'config false' "algorithms-supported" list to each of the algorithm-type modules.

A.14. 12 to 13

* Added the four features: "[encrypted-]one-[a]symmetric-key-format", each protecting a 'key-format' identity of the same name.

* Added 'must' expressions asserting that the 'key-format' leaf exists whenever a non-hidden key is specified.

* Improved the 'description' statements and added 'reference' statements for the 'key-format' identities.

* Added a questionable forward reference to "encrypted-*" leafs in a couple 'when' expressions.

* Did NOT move "config false" alg-supported lists to SSH/TLS drafts.

A.15. 13 to 14

* Resolved the "FIXME: forward ref" issue by modulating 'must', 'when', and 'mandatory' expressions.

* Moved the 'generatesymmetric-key' and 'generate-asymmetric-key' actions from ietf-keystore to ietf-crypto-types, now as RPCs.

* Cleaned up various description statements and removed lingering FIXMEs.

* Converted the "iana-<alg-type>-algs" YANG modules to IANA registries with instructions for how to generate modules from the registries, whenever they may be updated.

A.16. 14 to 15

* Removed the IANA-maintained registries for symmetric, asymmetric, and hash algorithms.
* Removed the "generate-symmetric-key" and "generate-asymmetric-key" RPCs.
* Removed the "algorithm" node in the various symmetric and asymmetric key groupings.
* Added 'typedef csr' and 'feature certificate-signing-request-generation'.
* Refined a usage of "end-entity-cert-grouping" to make the "cert" node mandatory true.
* Added a "Note to Reviewers" note to first page.

A.17. 15 to 16

* Updated draft title (refer to "Groupings" too).
* Removed 'end-entity-certs-grouping' as it wasn't being used anywhere.
* Removed 'trust-anchor-certs-grouping' as it was no longer being used after modifying 'local-or-truststore-certs-grouping' to use lists (not leaf-lists).
* Renamed "cert" to "cert-data" in trust-anchor-cert-grouping.
* Added "csr-info" typedef, to complement the existing "csr" typedef.
* Added "ocsp-request" and "ocsp-response" typedefs, to complement the existing "crl" typedef.
* Added "encrypted" cases to both symmetric-key-grouping and asymmetric-key-pair-grouping (Moved from Keystore draft).
* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
* Updated the Security Considerations section.

A.18. 16 to 17

* [Re]-added a "Strength of Keys Configured" Security Consideration

* Prefixed "cleartext-" in the "key" and "private-key" node names.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Balazs Kovacs, Eric Voit, Juergen Schoenwaelder, Liang Xia, Martin Bjorklund, Nick Hancock, Rich Salz, Rob Wilton, Tom Petch, and Wang Haiguang.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

YANG Data Types and Groupings for Cryptography
draft-ietf-netconf-crypto-types-34

Abstract

This document presents a YANG 1.1 (RFC 7950) module defining identities, typedefs, and groupings useful to cryptographic applications.

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 this draft

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

* 2024-03-16 --> the publication date of this draft

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

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on 17 September 2024.

Copyright Notice

Table of Contents

1. Introduction

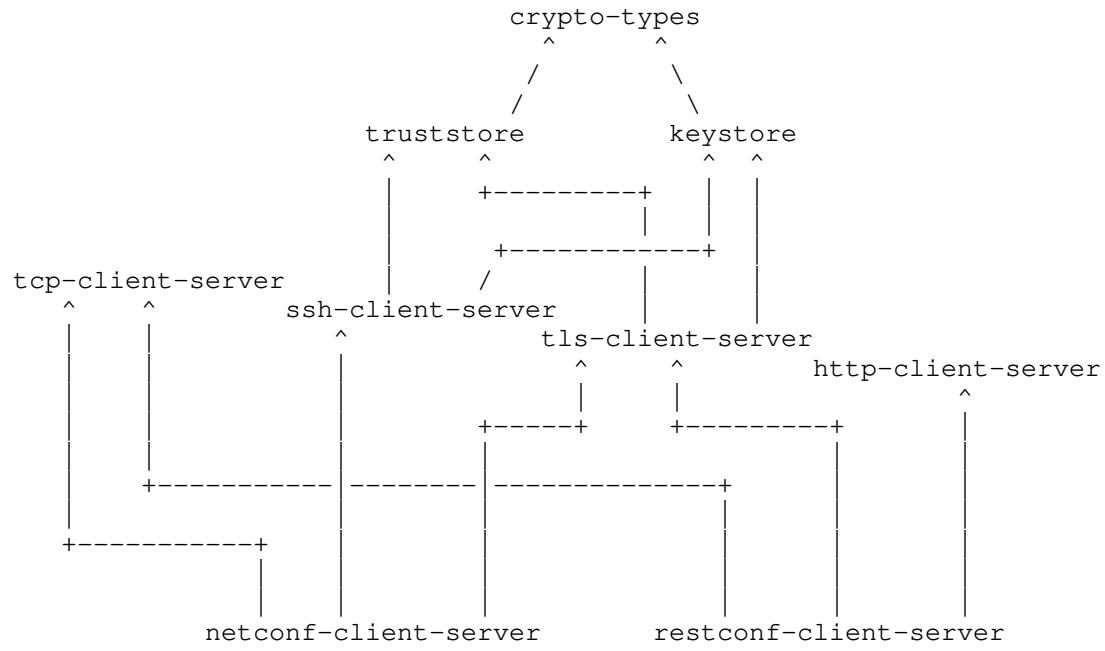
This document presents a YANG 1.1 [RFC7950] module defining identities, typedefs, and groupings useful to cryptographic applications.

1.1. Relation to other RFCs

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

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



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

Table 1: Label 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]. It does not define any protocol accessible nodes that are "config false".

1.4. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-crypto-types" Module

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

2.1. Data Model Overview

This section provides an overview of the "ietf-crypto-types" module in terms of its features, identities, typedefs, and groupings.

2.1.1. Features

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

```
   Features:
     +-- one-symmetric-key-format
     +-- one-asymmetric-key-format
     +-- symmetrically-encrypted-value-format
     +-- asymmetrically-encrypted-value-format
     +-- cms-enveloped-data-format
     +-- cms-encrypted-data-format
     +-- p10-csr-format
     +-- csr-generation
     +-- certificate-expiration-notification
     +-- cleartext-passwords
     +-- encrypted-passwords
     +-- cleartext-symmetric-keys
     +-- hidden-symmetric-keys
     +-- encrypted-symmetric-keys
     +-- cleartext-private-keys
     +-- hidden-private-keys
     +-- encrypted-private-keys
```

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

2.1.2. Identities

The following diagram illustrates the hierarchical relationship amongst the "identity" statements defined in the "ietf-crypto-types" module:

```
Identities:
  +-- public-key-format
  |  +-- subject-public-key-info-format
  |  +-- ssh-public-key-format
  +-- private-key-format
  |  +-- rsa-private-key-format
  |  +-- ec-private-key-format
  |  +-- one-asymmetric-key-format
  |             {one-asymmetric-key-format}?
  +-- symmetric-key-format
  |  +-- octet-string-key-format
  |  +-- one-symmetric-key-format
  |             {one-symmetric-key-format}?
  +-- encrypted-value-format
  |  +-- symmetrically-encrypted-value-format
  |  |  |       {symmetrically-encrypted-value-format}?
  |  |  +-- cms-encrypted-data-format
  |  |             {cms-encrypted-data-format}?
  |  +-- asymmetrically-encrypted-value-format
  |     |       {asymmetrically-encrypted-value-format}?
  |     +-- cms-enveloped-data-format
  |             {cms-enveloped-data-format}?
  +-- csr-format
        +-- p10-csr-format {p10-csr-format?}
```

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

Comments:

* The diagram shows that there are five base identities. The first three identities are used to indicate the format for the key data, while the fourth identity is used to indicate the format for encrypted values. The fifth identity is used to indicate the format for a certificate signing request. The base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of formats, rather than a specific format.

* The various terminal identities define specific encoding formats. The derived identities defined in this document are sufficient for the effort described in Section 1.1 but, by nature of them being identities, additional derived identities MAY be defined by future efforts.

* Identities used to specify uncommon formats are enabled by "feature" statements, allowing applications to support them when needed.

2.1.3. Typedefs

The following diagram illustrates the relationship amongst the "typedef" statements defined in the "ietf-crypto-types" module:

```
Typedefs:
  binary
    +-- csr-info
    +-- csr
    +-- x509
    |  +-- trust-anchor-cert-x509
    |  +-- end-entity-cert-x509
    +-- crl
    +-- ocsp-request
    +-- ocsp-response
    +-- cms
       +-- data-content-cms
       +-- signed-data-cms
       |  +-- trust-anchor-cert-cms
       |  +-- end-entity-cert-cms
       +-- enveloped-data-cms
       +-- digested-data-cms
       +-- encrypted-data-cms
       +-- authenticated-data-cms
```

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

Comments:

* All the typedefs defined in the "ietf-crypto-types" module extend the "binary" type defined in [RFC7950].

* Additionally, all the typedefs define a type for encoding an ASN.1 [ITU.X680.2021] structure using DER [ITU.X690.2021].

* The "trust-anchor-*" and "end-entity-*" typedefs are syntactically identical to their base typedefs and only distinguish themselves by the expected nature of their content. These typedefs are defined to facilitate common modeling needs.

2.1.4. Groupings

The "ietf-crypto-types" module defines the following "grouping" statements:

* encrypted-value-grouping
* password-grouping

* symmetric-key-grouping
* public-key-grouping
* private-key-grouping
* asymmetric-key-pair-grouping
* certificate-expiration-grouping
* trust-anchor-cert-grouping
* end-entity-cert-grouping
* generate-csr-grouping
* asymmetric-key-pair-with-cert-grouping
* asymmetric-key-pair-with-certs-grouping

Each of these groupings are presented in the following subsections.

2.1.4.1. The "encrypted-value-grouping" Grouping

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

```
  grouping encrypted-value-grouping:
    +-- encrypted-by
    +-- encrypted-value-format    identityref
    +-- encrypted-value           binary
```

Comments:

* The "encrypted-by" node is an empty container (difficult to see in the diagram) that a consuming module MUST augment key references into. The "ietf-crypto-types" module is unable to populate this container as the module only defines groupings. Section 2.2.1 presents an example illustrating a consuming module populating the "encrypted-by" container.

* The "encrypted-value" node is the value, encrypted by the key referenced by the "encrypted-by" node, and encoded in the format appropriate for the kind of key it was encrypted by.

  - If the value is encrypted by a symmetric key, then the encrypted value is encoded using the format associated with the "symmetrically-encrypted-value-format" identity.

  - If the value is encrypted by an asymmetric key, then the encrypted value is encoded using the format associated with the "asymmetrically-encrypted-value-format" identity.

  See Section 2.1.2 for information about the "format" identities.

2.1.4.2. The "password-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "password-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping password-grouping:
    +-- (password-type)
       +--:(cleartext-password) {cleartext-passwords}?
       |  +-- cleartext-password?   string
       +--:(encrypted-password) {encrypted-passwords}?
          +-- encrypted-password
             +---u encrypted-value-grouping
```

Comments:

* The "password-grouping" enables configuration of credentials needed to authenticate to a remote system. The 'ianach:crypt-hash' typedef from [RFC7317] should be used instead when needing to configure a password to authencate a local account.

* For the referenced grouping statement(s):

  - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "choice" statement enables the password data to be cleartext or encrypted, as follows:

  - The "cleartext-password" node can encode any cleartext value.
  - The "encrypted-password" node's structure is discussed in Section 2.1.4.1.

2.1.4.3. The "symmetric-key-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "symmetric-key-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping symmetric-key-grouping:
    +-- key-format?                       identityref
    +-- (key-type)
       +--:(cleartext-symmetric-key)
       |  +-- cleartext-symmetric-key?   binary
       |           {cleartext-symmetric-keys}?
       +--:(hidden-symmetric-key) {hidden-symmetric-keys}?
       |  +-- hidden-symmetric-key?      empty
       +--:(encrypted-symmetric-key) {encrypted-symmetric-keys}?
          +-- encrypted-symmetric-key
             +---u encrypted-value-grouping
```

Comments:

* For the referenced grouping statement(s):

  - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "key-format" node is an identity-reference to the "symmetric-key-format" abstract base identity discussed in Section 2.1.2, enabling the symmetric key to be encoded using any of the formats defined by the derived identities.

* The "choice" statement enables the private key data to be cleartext, encrypted, or hidden, as follows:

  - The "cleartext-symmetric-key" node can encode any cleartext key value.
  - The "hidden-symmetric-key" node is of type "empty" as the real value cannot be presented via the management interface.
  - The "encrypted-symmetric-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.4. The "public-key-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "public-key-grouping" grouping. This tree diagram does not expand any internally used grouping statement(s):

```
  grouping public-key-grouping:
    +-- public-key-format    identityref
    +-- public-key           binary
```

Comments:

* The "public-key-format" node is an identity-reference to the "public-key-format" abstract base identity discussed in Section 2.1.2, enabling the public key to be encoded using any of the formats defined by the derived identities.

* The "public-key" node is the public key data in the selected format. No "choice" statement is used to hide or encrypt the public key data because it is unnecessary to do so for public keys.

2.1.4.5. The "private-key-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "private-key-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping private-key-grouping:
    +-- private-key-format?            identityref
    +-- (private-key-type)
       +--:(cleartext-private-key) {cleartext-private-keys}?
       |  +-- cleartext-private-key?   binary
       +--:(hidden-private-key) {hidden-private-keys}?
       |  +-- hidden-private-key?      empty
       +--:(encrypted-private-key) {encrypted-private-keys}?
          +-- encrypted-private-key
             +---u encrypted-value-grouping
```

Comments:

* For the referenced grouping statement(s):

  - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "private-key-format" node is an identity-reference to the "private-key-format" abstract base identity discussed in Section 2.1.2, enabling the private key to be encoded using any of the formats defined by the derived identities.

* The "choice" statement enables the private key data to be cleartext, encrypted, or hidden, as follows:

  - The "cleartext-private-key" node can encode any cleartext key value.
  - The "hidden-private-key" node is of type "empty" as the real value cannot be presented via the management interface.
  - The "encrypted-private-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.6. The "asymmetric-key-pair-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "asymmetric-key-pair-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-grouping:
    +---u public-key-grouping
    +---u private-key-grouping
```

Comments:

* For the referenced grouping statement(s):

  - The "public-key-grouping" grouping is discussed in Section 2.1.4.4.
  - The "private-key-grouping" grouping is discussed in Section 2.1.4.5.

2.1.4.7. The "certificate-expiration-grouping" Grouping

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

```
  grouping certificate-expiration-grouping:
    +---n certificate-expiration
            {certificate-expiration-notification}?
       +-- expiration-date    yang:date-and-time
```

Comments:

* This grouping's only purpose is to define the "certificate-expiration" notification statement, used by the groupings defined in Section 2.1.4.8 and Section 2.1.4.9.

* The "certificate-expiration" notification enables servers to notify clients when certificates are nearing expiration.

* The "expiration-date" node indicates when the designated certificate will (or did) expire.

* Identification of the certificate that is expiring is built into the notification itself. For an example, please see Section 2.2.3.

2.1.4.8. The "trust-anchor-cert-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "trust-anchor-cert-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping trust-anchor-cert-grouping:
    +-- cert-data?                          trust-anchor-cert-cms
    +---u certificate-expiration-grouping
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.7.

* The "cert-data" node contains a chain of one or more certificates containing at most one self-signed certificates (the "root" certificate), encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.9. The "end-entity-cert-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "end-entity-cert-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping end-entity-cert-grouping:
    +-- cert-data?                          end-entity-cert-cms
    +---u certificate-expiration-grouping
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.7.

* The "cert-data" node contains a chain of one or more certificates containing at most one certificate that is neither self-signed nor having Basic constraint "CA true", encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.10. The "generate-csr-grouping" Grouping

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

```
  grouping generate-csr-grouping:
    +---x generate-csr {csr-generation}?
       +---w input
       |  +---w csr-format    identityref
       |  +---w csr-info      csr-info
       +--ro output
          +--ro (csr-type)
             +--:(p10-csr)
                +--ro p10-csr?   p10-csr
```

Comments:

* This grouping's only purpose is to define the "generate-certificate-signing-request" action statement, used by the groupings defined in Section 2.1.4.11 and Section 2.1.4.12.

* This action takes as input a "csr-info" type and returns a "csr" type, both of which are discussed in Section 2.1.3.

* For an example, please see Section 2.2.2.

2.1.4.11. The "asymmetric-key-pair-with-cert-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "asymmetric-key-pair-with-cert-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-with-cert-grouping:
    +---u asymmetric-key-pair-grouping
    +---u end-entity-cert-grouping
    +---u generate-csr-grouping
```

Comments:

* This grouping defines an asymmetric key with at most one associated certificate, a commonly needed combination in protocol models.

* For the referenced grouping statement(s):

  - The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.6.
  - The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.9.
  - The "generate-csr-grouping" grouping is discussed in Section 2.1.4.10.

2.1.4.12. The "asymmetric-key-pair-with-certs-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "asymmetric-key-pair-with-certs-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-with-certs-grouping:
    +---u asymmetric-key-pair-grouping
    +-- certificates
    |  +-- certificate* [name]
    |     +-- name?                        string
    |     +---u end-entity-cert-grouping
    +---u generate-csr-grouping
```

Comments:

* This grouping defines an asymmetric key with one or more associated certificates, a commonly needed combination in configuration models.

* For the referenced grouping statement(s):

  - The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.6.
  - The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.9.
  - The "generate-csr-grouping" grouping is discussed in Section 2.1.4.10.

2.1.5. Protocol-accessible Nodes

The "ietf-crypto-types" module does not contain any protocol-accessible nodes, but the module needs to be "implemented", as described in Section 5.6.5 of [RFC7950], in order for the identities in Section 2.1.2 to be defined.

2.2. Example Usage

2.2.1. The "symmetric-key-grouping", "asymmetric-key-pair-with-certs-grouping", and "password-grouping" Groupings

The following non-normative module is constructed in order to illustrate the use of the "symmetric-key-grouping" (Section 2.1.4.3), the "asymmetric-key-pair-with-certs-grouping" (Section 2.1.4.12), and the "password-grouping" (Section 2.1.4.2) grouping statements.

Notably, this example module and associated configuration data illustrates that a hidden private key (ex-hidden-asymmetric-key) has been used to encrypt a symmetric key (ex-encrypted-one-symmetric-based-symmetric-key) that has been used to encrypt another private key (ex-encrypted-rsa-based-asymmetric-key). Additionally, the symmetric key is also used to encrypt a password (ex-encrypted-password).

2.2.1.1. Example Module

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

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

  organization
    "Example Corporation";
  contact
    "YANG Designer <mailto:yang.designer@example.com>";

  description
    "This example module illustrates the 'symmetric-key-grouping'
     and 'asymmetric-key-grouping' groupings defined in the
    'ietf-crypto-types' module defined in RFC AAAA.";

  revision 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC AAAA: Common YANG Data Types for Cryptography";
  }

  container symmetric-keys {
    description
      "A container of symmetric keys.";
    list symmetric-key {
      key "name";
      description
        "A symmetric key";
      leaf name {
        type string;
        description
```

```
            "An arbitrary name for this key.";
          }
          uses ct:symmetric-key-grouping {
            augment "key-type/encrypted-symmetric-key/"
                  + "encrypted-symmetric-key/encrypted-by" {
              description
                "Augments in a choice statement enabling the
                 encrypting key to be any other symmetric or
                 asymmetric key.";
              uses encrypted-by-grouping;
            }
          }
        }
      }
      container asymmetric-keys {
        description
          "A container of asymmetric keys.";
        list asymmetric-key {
          key "name";
          leaf name {
            type string;
            description
              "An arbitrary name for this key.";
          }
          uses ct:asymmetric-key-pair-with-certs-grouping {
            augment "private-key-type/encrypted-private-key/"
                  + "encrypted-private-key/encrypted-by" {
              description
                "Augments in a choice statement enabling the
                 encrypting key to be any other symmetric or
                 asymmetric key.";
              uses encrypted-by-grouping;
            }
          }
          description
            "An asymmetric key pair with associated certificates.";
        }
      }
      container passwords {
        description
          "A container of passwords.";
        list password {
          key "name";
          leaf name {
            type string;
            description
              "An arbitrary name for this password.";
          }
```

```
        uses ct:password-grouping {
          augment "password-type/encrypted-password/"
                + "encrypted-password/encrypted-by" {
            description
              "Augments in a choice statement enabling the
               encrypting key to be any symmetric or
               asymmetric key.";
            uses encrypted-by-grouping;
          }
        }
        description
          "A password.";
      }
    }

    grouping encrypted-by-grouping {
      description
        "A grouping that defines a choice enabling references
         to other keys.";
      choice encrypted-by {
        mandatory true;
        description
          "A choice amongst other symmetric or asymmetric keys.";
        case symmetric-key-ref {
          leaf symmetric-key-ref {
            type leafref {
              path "/ectu:symmetric-keys/ectu:symmetric-key/"
                 + "ectu:name";
            }
            description
              "Identifies the symmetric key that encrypts this key.";
          }
        }
        case asymmetric-key-ref {
          leaf asymmetric-key-ref {
            type leafref {
              path "/ectu:asymmetric-keys/ectu:asymmetric-key/"
                 + "ectu:name";
            }
            description
              "Identifies the asymmetric key that encrypts this key.";
          }
        }
      }
    }
  }
```

2.2.1.2. Tree Diagram for the Example Module

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

```
module: ex-crypto-types-usage
  +--rw symmetric-keys
  |  +--rw symmetric-key* [name]
  |     +--rw name                                string
  |     +--rw key-format?                         identityref
  |     +--rw (key-type)
  |        +--:(cleartext-symmetric-key)
  |        |  +--rw cleartext-symmetric-key?   binary
  |        |          {cleartext-symmetric-keys}?
  |        +--:(hidden-symmetric-key) {hidden-symmetric-keys}?
  |        |  +--rw hidden-symmetric-key?      empty
  |        +--:(encrypted-symmetric-key) {encrypted-symmetric-keys}?
  |           +--rw encrypted-symmetric-key
  |              +--rw encrypted-by
  |              |  +--rw (encrypted-by)
  |              |     +--:(symmetric-key-ref)
  |              |     |  +--rw symmetric-key-ref?    leafref
  |              |     +--:(asymmetric-key-ref)
  |              |        +--rw asymmetric-key-ref?   leafref
  |              +--rw encrypted-value-format    identityref
  |              +--rw encrypted-value           binary
  +--rw asymmetric-keys
  |  +--rw asymmetric-key* [name]
  |     +--rw name                              string
  |     +--rw public-key-format?                identityref
  |     +--rw public-key?                       binary
  |     +--rw private-key-format?               identityref
  |     +--rw (private-key-type)
  |     |  +--:(cleartext-private-key) {cleartext-private-keys}?
  |     |  |  +--rw cleartext-private-key?   binary
  |     |  +--:(hidden-private-key) {hidden-private-keys}?
  |     |  |  +--rw hidden-private-key?      empty
  |     |  +--:(encrypted-private-key) {encrypted-private-keys}?
  |     |     +--rw encrypted-private-key
  |     |        +--rw encrypted-by
  |     |        |  +--rw (encrypted-by)
  |     |        |     +--:(symmetric-key-ref)
  |     |        |     |  +--rw symmetric-key-ref?    leafref
  |     |        |     +--:(asymmetric-key-ref)
  |     |        |        +--rw asymmetric-key-ref?   leafref
  |     |        +--rw encrypted-value-format    identityref
  |     |        +--rw encrypted-value           binary
  |     +--rw certificates
  |     |  +--rw certificate* [name]
```

```
      |      |     +--rw name                       string
      |      |     +--rw cert-data                  end-entity-cert-cms
      |      |     +---n certificate-expiration
      |      |             {certificate-expiration-notification}?
      |      |        +-- expiration-date    yang:date-and-time
      |      +---x generate-csr {csr-generation}?
      |         +---w input
      |         |  +---w csr-format    identityref
      |         |  +---w csr-info      csr-info
      |         +--ro output
      |            +--ro (csr-type)
      |               +--:(p10-csr)
      |                  +--ro p10-csr?   p10-csr
      +--rw passwords
         +--rw password* [name]
            +--rw name                          string
            +--rw (password-type)
               +--:(cleartext-password) {cleartext-passwords}?
               |  +--rw cleartext-password?   string
               +--:(encrypted-password) {encrypted-passwords}?
                  +--rw encrypted-password
                     +--rw encrypted-by
                     |  +--rw (encrypted-by)
                     |     +--:(symmetric-key-ref)
                     |     |  +--rw symmetric-key-ref?    leafref
                     |     +--:(asymmetric-key-ref)
                     |        +--rw asymmetric-key-ref?   leafref
                     +--rw encrypted-value-format    identityref
                     +--rw encrypted-value           binary
```

2.2.1.3. Usage Example for the Example Module

Finally, the following example illustrates various symmetric and asymmetric keys as they might appear in configuration:

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

<symmetric-keys
  xmlns="https://example.com/ns/example-crypto-types-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <symmetric-key>
    <name>ex-hidden-symmetric-key</name>
    <hidden-symmetric-key/>
  </symmetric-key>
  <symmetric-key>
    <name>ex-octet-string-based-symmetric-key</name>
    <key-format>ct:octet-string-key-format</key-format>
    <cleartext-symmetric-key>BASE64VALUE=</cleartext-symmetric-key>
```

```
    </symmetric-key>
    <symmetric-key>
      <name>ex-one-symmetric-based-symmetric-key</name>
      <key-format>ct:one-symmetric-key-format</key-format>
      <cleartext-symmetric-key>BASE64VALUE=</cleartext-symmetric-key>
    </symmetric-key>
    <symmetric-key>
      <name>ex-encrypted-one-symmetric-based-symmetric-key</name>
      <key-format>ct:one-symmetric-key-format</key-format>
      <encrypted-symmetric-key>
        <encrypted-by>
          <asymmetric-key-ref>ex-hidden-asymmetric-key</asymmetric-key\
-ref>
        </encrypted-by>
        <encrypted-value-format>ct:cms-enveloped-data-format</encrypte\
d-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-symmetric-key>
    </symmetric-key>
  </symmetric-keys>

  <asymmetric-keys
    xmlns="https://example.com/ns/example-crypto-types-usage"
    xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <asymmetric-key>
      <name>ex-hidden-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>ex-hidden-asymmetric-key-cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-rsa-based-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:rsa-private-key-format</private-key-forma\
t>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      <certificates>
        <certificate>
          <name>ex-cert</name>
```

```
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-one-asymmetric-based-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:one-asymmetric-key-format</private-key-fo\
rmat>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-encrypted-rsa-based-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:rsa-private-key-format</private-key-forma\
t>
      <encrypted-private-key>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
        </encrypted-by>
        <encrypted-value-format>ct:cms-encrypted-data-format</encrypte\
d-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-private-key>
    </asymmetric-key>
  </asymmetric-keys>

  <passwords
    xmlns="https://example.com/ns/example-crypto-types-usage"
    xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <password>
      <name>ex-cleartext-password</name>
      <cleartext-password>super-secret</cleartext-password>
    </password>
    <password>
      <name>ex-encrypted-password</name>
      <encrypted-password>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
        </encrypted-by>
        <encrypted-value-format>ct:cms-encrypted-data-format</encrypte\
d-value-format>
```

```
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-password>
    </password>
  </passwords>
```

2.2.2. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action, discussed in Section 2.1.4.10, with the NETCONF protocol.

REQUEST

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <asymmetric-keys
      xmlns="https://example.com/ns/example-crypto-types-usage">
      <asymmetric-key>
        <name>ex-hidden-asymmetric-key</name>
        <generate-csr>
          <csr-format>ct:p10-csr-format</csr-format>
          <csr-info>BASE64VALUE=</csr-info>
        </generate-csr>
      </asymmetric-key>
    </asymmetric-keys>
  </action>
</rpc>
```

RESPONSE

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

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <p10-csr xmlns="https://example.com/ns/example-crypto-types-usage"\
>BASE64VALUE=</p10-csr>
</rpc-reply>
```

2.2.3. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification, discussed in Section 2.1.4.7, with the NETCONF protocol.

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

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <asymmetric-keys xmlns="https://example.com/ns/example-crypto-type\
   s-usage">
       <asymmetric-key>
         <name>ex-hidden-asymmetric-key</name>
         <certificates>
           <certificate>
             <name>ex-hidden-asymmetric-key-cert</name>
             <certificate-expiration>
               <expiration-date>2018-08-05T14:18:53-05:00</expiration-d\
   ate>
             </certificate-expiration>
           </certificate>
         </certificates>
       </asymmetric-key>
     </asymmetric-keys>
   </notification>
```

2.3. YANG Module

This module has normative references to [RFC2119], [RFC2986], [RFC4253], [RFC5280], [RFC5652], [RFC5915], [RFC5958], [RFC6031], [RFC6960], [RFC6991], [RFC7093], [RFC8017], [RFC8174], [RFC8341], and [ITU.X690.2021].

```
   <CODE BEGINS> file "ietf-crypto-types@2024-03-16.yang"

   module ietf-crypto-types {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-crypto-types";
     prefix ct;

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

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

```
  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module defines common YANG types for cryptographic
     applications.

     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 AAAA
     (https://www.rfc-editor.org/info/rfcAAAA); 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 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

  feature one-symmetric-key-format {
    description
      "Indicates that the server supports the
       'one-symmetric-key-format' identity.";
```

```
  }

  feature one-asymmetric-key-format {
    description
      "Indicates that the server supports the
       'one-asymmetric-key-format' identity.";
  }

  feature symmetrically-encrypted-value-format {
    description
      "Indicates that the server supports the
       'symmetrically-encrypted-value-format' identity.";
  }

  feature asymmetrically-encrypted-value-format {
    description
      "Indicates that the server supports the
       'asymmetrically-encrypted-value-format' identity.";
  }

  feature cms-enveloped-data-format {
    description
      "Indicates that the server supports the
       'cms-enveloped-data-format' identity.";
  }

  feature cms-encrypted-data-format {
    description
      "Indicates that the server supports the
       'cms-encrypted-data-format' identity.";
  }

  feature p10-csr-format {
    description
      "Indicates that the server implements support
       for generating P10-based CSRs, as defined
       in RFC 2986.";
    reference
      "RFC 2986: PKCS #10: Certification Request Syntax
                 Specification Version 1.7";
  }

  feature csr-generation {
    description
      "Indicates that the server implements the
       'generate-csr' action.";
  }
```

```
    feature certificate-expiration-notification {
      description
        "Indicates that the server implements the
         'certificate-expiration' notification.";
    }

    feature cleartext-passwords {
      description
        "Indicates that the server supports cleartext
         passwords.";
    }

    feature encrypted-passwords {
      description
        "Indicates that the server supports password
         encryption.";
    }

    feature cleartext-symmetric-keys {
      description
        "Indicates that the server supports cleartext
         symmetric keys.";
    }

    feature hidden-symmetric-keys {
      description
        "Indicates that the server supports hidden keys.";
    }

    feature encrypted-symmetric-keys {
      description
        "Indicates that the server supports encryption
         of symmetric keys.";
    }

    feature cleartext-private-keys {
      description
        "Indicates that the server supports cleartext
         private keys.";
    }

    feature hidden-private-keys {
      description
        "Indicates that the server supports hidden keys.";
    }

    feature encrypted-private-keys {
      description
```

```
        "Indicates that the server supports encryption
         of private keys.";
    }

    /************************************************/
    /*   Base Identities for Key Format Structures   */
    /************************************************/

    identity symmetric-key-format {
      description
        "Base key-format identity for symmetric keys.";
    }

    identity public-key-format {
      description
        "Base key-format identity for public keys.";
    }

    identity private-key-format {
      description
        "Base key-format identity for private keys.";
    }

    /*****************************************************/
    /*   Identities for Private Key Format Structures   */
    /*****************************************************/

    identity rsa-private-key-format {
      base private-key-format;
      description
        "Indicates that the private key value is encoded as
         an RSAPrivateKey (from RFC 8017), encoded using ASN.1
         distinguished encoding rules (DER), as specified in
         ITU-T X.690.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER) 02/2021.";
    }

    identity ec-private-key-format {
      base private-key-format;
      description
        "Indicates that the private key value is encoded as
```

```
          an ECPrivateKey (from RFC 5915), encoded using ASN.1
          distinguished encoding rules (DER), as specified in
          ITU-T X.690.";
        reference
          "RFC 5915:
            Elliptic Curve Private Key Structure
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER) 02/2021.";
      }

      identity one-asymmetric-key-format {
        if-feature "one-asymmetric-key-format";
        base private-key-format;
        description
          "Indicates that the private key value is a CMS
           OneAsymmetricKey structure, as defined in RFC 5958,
           encoded using ASN.1 distinguished encoding rules
           (DER), as specified in ITU-T X.690.";
        reference
          "RFC 5958: Asymmetric Key Packages
           ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER) 02/2021.";
      }

      /*************************************************/
      /*   Identities for Public Key Format Structures   */
      /*************************************************/

      identity ssh-public-key-format {
        base public-key-format;
        description
          "Indicates that the public key value is an SSH public key,
           as specified by RFC 4253, Section 6.6, i.e.:

            string    certificate or public key format
                      identifier
            byte[n]   key/certificate data.";
        reference
          "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      }

      identity subject-public-key-info-format {
```

```
      base public-key-format;
      description
        "Indicates that the public key value is a SubjectPublicKeyInfo
         structure, as described in RFC 5280 encoded using ASN.1
         distinguished encoding rules (DER), as specified in
         ITU-T X.690.";
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER) 02/2021.";
    }

    /*****************************************************/
    /*   Identities for Symmetric Key Format Structures   */
    /*****************************************************/

    identity octet-string-key-format {
      base symmetric-key-format;
      description
        "Indicates that the key is encoded as a raw octet string.
         The length of the octet string MUST be appropriate for
         the associated algorithm's block size.

          The identity of the associated algorithm is outside the
          scope of this specification.  This is also true when
          the octet string has been encrypted.";
    }

    identity one-symmetric-key-format {
      if-feature "one-symmetric-key-format";
      base symmetric-key-format;
      description
        "Indicates that the private key value is a CMS
         OneSymmetricKey structure, as defined in RFC 6031,
         encoded using ASN.1 distinguished encoding rules
         (DER), as specified in ITU-T X.690.";
      reference
        "RFC 6031: Cryptographic Message Syntax (CMS)
                   Symmetric Key Package Content Type
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
```

```
          Encoding Rules (DER) 02/2021.";
    }

    /************************************************/
    /*   Identities for Encrypted Value Structures   */
    /************************************************/

    identity encrypted-value-format {
      description
        "Base format identity for encrypted values.";
    }

    identity symmetrically-encrypted-value-format {
      if-feature "symmetrically-encrypted-value-format";
      base encrypted-value-format;
      description
        "Base format identity for symmetrically encrypted
         values.";
    }

    identity asymmetrically-encrypted-value-format {
      if-feature "asymmetrically-encrypted-value-format";
      base encrypted-value-format;
      description
        "Base format identity for asymmetrically encrypted
         values.";
    }

    identity cms-encrypted-data-format {
      if-feature "cms-encrypted-data-format";
      base symmetrically-encrypted-value-format;
      description
        "Indicates that the encrypted value conforms to
         the 'encrypted-data-cms' type with the constraint
         that the 'unprotectedAttrs' value is not set.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER) 02/2021.";
    }

    identity cms-enveloped-data-format {
      if-feature "cms-enveloped-data-format";
      base asymmetrically-encrypted-value-format;
      description
```

```
        "Indicates that the encrypted value conforms to the
         'enveloped-data-cms' type with the following constraints:

         The EnvelopedData structure MUST have exactly one
         'RecipientInfo'.

         If the asymmetric key supports public key cryptography
         (e.g., RSA), then the 'RecipientInfo' must be a
         'KeyTransRecipientInfo' with the 'RecipientIdentifier'
         using a 'subjectKeyIdentifier' with the value set using
         'method 1' in RFC 7093 over the recipient's public key.

         Otherwise, if the asymmetric key supports key agreement
         (e.g., ECC), then the 'RecipientInfo' must be a
         'KeyAgreeRecipientInfo'.  The 'OriginatorIdentifierOrKey'
         value must use the 'OriginatorPublicKey' alternative.
         The 'UserKeyingMaterial' value must not be present.
         There must be exactly one 'RecipientEncryptedKeys' value
         having the 'KeyAgreeRecipientIdentifier' set to 'rKeyId'
         with the value set using 'method 1' in RFC 7093 over the
         recipient's public key.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)
         RFC 7093:
           Additional Methods for Generating Key
           Identifiers Values
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER) 02/2021.";
    }

    /*****************************************************/
    /*   Identities for Certificate Signing Request Formats  */
    /*****************************************************/

    identity csr-format {
      description
        "A base identity for the certificate signing request
         formats.  Additional derived identities MAY be defined
         by future efforts.";
    }

    identity p10-csr-format {
      if-feature "p10-csr-format";
      base csr-format;
      description
```

```
        "Indicates the 'CertificationRequest' structure
         defined in RFC 2986.";
      reference
        "RFC 2986: PKCS #10: Certification Request Syntax
                   Specification Version 1.7";
    }


    /***************************************************/
    /*   Typedefs for ASN.1 structures from RFC 2986   */
    /***************************************************/

    typedef csr-info {
      type binary;
      description
        "A CertificationRequestInfo structure, as defined in
         RFC 2986, encoded using ASN.1 distinguished encoding
         rules (DER), as specified in ITU-T X.690.";
      reference
        "RFC 2986: PKCS #10: Certification Request Syntax
                   Specification Version 1.7
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER) 02/2021.";
    }

    typedef p10-csr {
      type binary;
      description
        "A CertificationRequest structure, as specified in
         RFC 2986, encoded using ASN.1 distinguished encoding
         rules (DER), as specified in ITU-T X.690.";
      reference
        "RFC 2986:
           PKCS #10: Certification Request Syntax Specification
           Version 1.7
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER) 02/2021.";
    }

    /***************************************************/
    /*   Typedefs for ASN.1 structures from RFC 5280   */
    /***************************************************/
```

```
     typedef x509 {
       type binary;
       description
         "A Certificate structure, as specified in RFC 5280,
          encoded using ASN.1 distinguished encoding rules (DER),
          as specified in ITU-T X.690.";
       reference
         "RFC 5280:
            Internet X.509 Public Key Infrastructure Certificate
            and Certificate Revocation List (CRL) Profile
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER) 02/2021.";
     }

     typedef crl {
       type binary;
       description
         "A CertificateList structure, as specified in RFC 5280,
          encoded using ASN.1 distinguished encoding rules (DER),
          as specified in ITU-T X.690.";
       reference
         "RFC 5280:
            Internet X.509 Public Key Infrastructure Certificate
            and Certificate Revocation List (CRL) Profile
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER) 02/2021.";
     }

     /*************************************************/
     /*   Typedefs for ASN.1 structures from RFC 6960   */
     /*************************************************/

     typedef oscp-request {
       type binary;
       description
         "A OCSPRequest structure, as specified in RFC 6960,
          encoded using ASN.1 distinguished encoding rules
          (DER), as specified in ITU-T X.690.";
       reference
         "RFC 6960:
            X.509 Internet Public Key Infrastructure Online
            Certificate Status Protocol - OCSP
```

```
          ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER) 02/2021.";
      }

      typedef oscp-response {
        type binary;
        description
          "A OCSPResponse structure, as specified in RFC 6960,
           encoded using ASN.1 distinguished encoding rules
           (DER), as specified in ITU-T X.690.";
        reference
          "RFC 6960:
             X.509 Internet Public Key Infrastructure Online
             Certificate Status Protocol - OCSP
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER) 02/2021.";
      }

      /*********************************************/
      /*   Typedefs for ASN.1 structures from 5652   */
      /*********************************************/

      typedef cms {
        type binary;
        description
          "A ContentInfo structure, as specified in RFC 5652,
           encoded using ASN.1 distinguished encoding rules (DER),
           as specified in ITU-T X.690.";
        reference
          "RFC 5652:
             Cryptographic Message Syntax (CMS)
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER) 02/2021.";
      }

      typedef data-content-cms {
        type cms;
        description
          "A CMS structure whose top-most content type MUST be the
```

```
          data content type, as described by Section 4 in RFC 5652.";
        reference
          "RFC 5652: Cryptographic Message Syntax (CMS)";
      }

      typedef signed-data-cms {
        type cms;
        description
          "A CMS structure whose top-most content type MUST be the
           signed-data content type, as described by Section 5 in
           RFC 5652.";
        reference
          "RFC 5652: Cryptographic Message Syntax (CMS)";
      }

      typedef enveloped-data-cms {
        type cms;
        description
          "A CMS structure whose top-most content type MUST be the
           enveloped-data content type, as described by Section 6
           in RFC 5652.";
        reference
          "RFC 5652: Cryptographic Message Syntax (CMS)";
      }

      typedef digested-data-cms {
        type cms;
        description
          "A CMS structure whose top-most content type MUST be the
           digested-data content type, as described by Section 7
           in RFC 5652.";
        reference
          "RFC 5652: Cryptographic Message Syntax (CMS)";
      }

      typedef encrypted-data-cms {
        type cms;
        description
          "A CMS structure whose top-most content type MUST be the
           encrypted-data content type, as described by Section 8
           in RFC 5652.";
        reference
          "RFC 5652: Cryptographic Message Syntax (CMS)";
      }

      typedef authenticated-data-cms {
        type cms;
        description
```

```
        "A CMS structure whose top-most content type MUST be the
         authenticated-data content type, as described by Section 9
         in RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    /*****************************************************/
    /*   Typedefs for ASN.1 structures related to RFC 5280   */
    /*****************************************************/

    typedef trust-anchor-cert-x509 {
      type x509;
      description
        "A Certificate structure that MUST encode a self-signed
         root certificate.";
    }

    typedef end-entity-cert-x509 {
      type x509;
      description
        "A Certificate structure that MUST encode a certificate
         that is neither self-signed nor having Basic constraint
         CA true.";
    }

    /*****************************************************/
    /*   Typedefs for ASN.1 structures related to RFC 5652   */
    /*****************************************************/

    typedef trust-anchor-cert-cms {
      type signed-data-cms;
      description
        "A CMS SignedData structure that MUST contain the chain of
         X.509 certificates needed to authenticate the certificate
         presented by a client or end-entity.

         The CMS MUST contain only a single chain of certificates.
         The client or end-entity certificate MUST only authenticate
         to the last intermediate CA certificate listed in the chain.

         In all cases, the chain MUST include a self-signed root
         certificate.  In the case where the root certificate is
         itself the issuer of the client or end-entity certificate,
         only one certificate is present.

         This CMS structure MAY (as applicable where this type is
         used) also contain suitably fresh (as defined by local
```

```
          policy) revocation objects with which the device can
          verify the revocation status of the certificates.

          This CMS encodes the degenerate form of the SignedData
          structure (RFC 5652, Section 5.2) that is commonly used
          to disseminate X.509 certificates and revocation objects
          (RFC 5280).";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile.
           RFC 5652:
             Cryptographic Message Syntax (CMS)";
      }

      typedef end-entity-cert-cms {
        type signed-data-cms;
        description
          "A CMS SignedData structure that MUST contain the end
           entity certificate itself, and MAY contain any number
           of intermediate certificates leading up to a trust
           anchor certificate.  The trust anchor certificate
           MAY be included as well.

           The CMS MUST contain a single end entity certificate.
           The CMS MUST NOT contain any spurious certificates.

           This CMS structure MAY (as applicable where this type is
           used) also contain suitably fresh (as defined by local
           policy) revocation objects with which the device can
           verify the revocation status of the certificates.

           This CMS encodes the degenerate form of the SignedData
           structure (RFC 5652, Section 5.2) that is commonly
           used to disseminate X.509 certificates and revocation
           objects (RFC 5280).";

        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile.
           RFC 5652:
             Cryptographic Message Syntax (CMS)";
      }

      /*****************/
      /*   Groupings   */
      /*****************/
```

```
  grouping encrypted-value-grouping {
    description
      "A reusable grouping for a value that has been encrypted by
       a referenced symmetric or asymmetric key.";
    container encrypted-by {
      nacm:default-deny-write;
      description
        "An empty container enabling a reference to the key that
         encrypted the value to be augmented in.  The referenced
         key MUST be a symmetric key or an asymmetric key.

         A symmetric key MUST be referenced via a leaf node called
         'symmetric-key-ref'.  An asymmetric key MUST be referenced
         via a leaf node called 'asymmetric-key-ref'.

         The leaf nodes MUST be direct descendants in the data tree,
         and MAY be direct descendants in the schema tree (e.g.,
         choice/case statements are allowed, but not a container).";
    }
    leaf encrypted-value-format {
      type identityref {
        base encrypted-value-format;
      }
      mandatory true;
      description
        "Identifies the format of the 'encrypted-value' leaf.

         If 'encrypted-by' points to a symmetric key, then a
         'symmetrically-encrypted-value-format' based identity
         MUST be set (e.g., cms-encrypted-data-format).

         If 'encrypted-by' points to an asymmetric key, then an
         'asymmetrically-encrypted-value-format' based identity
         MUST be set (e.g., cms-enveloped-data-format).";
    }
    leaf encrypted-value {
      nacm:default-deny-write;
      type binary;
      must '../encrypted-by';
      mandatory true;
      description
        "The value, encrypted using the referenced symmetric
         or asymmetric key.  The value MUST be encoded using
         the format associated with the 'encrypted-value-format'
         leaf.";
    }
  }
```

```
     grouping password-grouping {
       description
         "A password used for authenticating to a remote system.

          The 'ianach:crypt-hash' typedef from RFC 7317 should be
          used instead when needing a password to authencate a
          local account.";
       choice password-type {
         nacm:default-deny-write;
         mandatory true;
         description
           "Choice between password types.";
         case cleartext-password {
           if-feature "cleartext-passwords";
           leaf cleartext-password {
             nacm:default-deny-all;
             type string;
             description
               "The cleartext value of the password.";
           }
         }
         case encrypted-password {
           if-feature "encrypted-passwords";
           container encrypted-password {
             description
               "A container for the encrypted password value.";
             uses encrypted-value-grouping;
           }
         }
       }
     }

     grouping symmetric-key-grouping {
       description
         "A symmetric key.";
       leaf key-format {
         nacm:default-deny-write;
         type identityref {
           base symmetric-key-format;
         }
         description
           "Identifies the symmetric key's format.  Implementations
            SHOULD ensure that the incoming symmetric key value is
            encoded in the specified format.

            For encrypted keys, the value is the decrypted key's
            format (i.e., the 'encrypted-value-format' conveys the
            encrypted key's format.";
```

```
      }
      choice key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-symmetric-key {
          leaf cleartext-symmetric-key {
            if-feature "cleartext-symmetric-keys";
            nacm:default-deny-all;
            type binary;
            must '../key-format';
            description
              "The binary value of the key.  The interpretation of
               the value is defined by the 'key-format' field.";
          }
        }
        case hidden-symmetric-key {
          if-feature "hidden-symmetric-keys";
          leaf hidden-symmetric-key {
            type empty;
            must 'not(../key-format)';
            description
              "A hidden key is not exportable, and not extractable,
               and therefore, it is of type 'empty' as its value is
               inaccessible via management interfaces. Though hidden
               to users, such keys are not hidden to the server and
               may be referenced by configuration to indicate which
               key a server should use for a cryptographic operation.
               How such keys are created is outside the scope of this
               module.";
          }
        }
        case encrypted-symmetric-key {
          if-feature "encrypted-symmetric-keys";
          container encrypted-symmetric-key {
            must '../key-format';
            description
              "A container for the encrypted symmetric key value.
               The interpretation of the 'encrypted-value' node
               is via the 'key-format' node";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping public-key-grouping {
```

```
      description
        "A public key.";
      leaf public-key-format {
        nacm:default-deny-write;
        type identityref {
          base public-key-format;
        }
        mandatory true;
        description
          "Identifies the public key's format. Implementations SHOULD
           ensure that the incoming public key value is encoded in the
           specified format.";
      }
      leaf public-key {
        nacm:default-deny-write;
        type binary;
        mandatory true;
        description
          "The binary value of the public key.  The interpretation
           of the value is defined by 'public-key-format' field.";
      }
    }

    grouping private-key-grouping {
      description
        "A private key.";
      leaf private-key-format {
        nacm:default-deny-write;
        type identityref {
          base private-key-format;
        }
        description
          "Identifies the private key's format.  Implementations SHOULD
           ensure that the incoming private key value is encoded in the
           specified format.

           For encrypted keys, the value is the decrypted key's
           format (i.e., the 'encrypted-value-format' conveys the
           encrypted key's format.";
      }
      choice private-key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-private-key {
          if-feature "cleartext-private-keys";
          leaf cleartext-private-key {
```

```
            nacm:default-deny-all;
            type binary;
            must '../private-key-format';
            description
              "The value of the binary key  The key's value is
               interpreted by the 'private-key-format' field.";
          }
        }
        case hidden-private-key {
          if-feature "hidden-private-keys";
          leaf hidden-private-key {
            type empty;
            must 'not(../private-key-format)';
            description
              "A hidden key.  It is of type 'empty' as its value is
               inaccessible via management interfaces.  Though hidden
               to users, such keys are not hidden to the server and
               and may be referenced by configuration to indicate which
               key a server should use for a cryptographic operation.
               How such keys are created is outside the scope of this
               module.";
          }
        }
        case encrypted-private-key {
          if-feature "encrypted-private-keys";
          container encrypted-private-key {
            must '../private-key-format';
            description
              "A container for the encrypted asymmetric private key
               value.  The interpretation of the 'encrypted-value'
               node is via the 'private-key-format' node";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping asymmetric-key-pair-grouping {
      description
        "A private key and, optionally, its associated public key.
         Implementations MUST ensure that the two keys, when both
         are specified, are a matching pair.";
      uses public-key-grouping {
        refine public-key-format {
          mandatory false;
        }
        refine public-key {
          mandatory false;
```

```
        }
      }
      uses private-key-grouping;
    }

    grouping certificate-expiration-grouping {
      description
        "A notification for when a certificate is about to, or
         already has, expired.";
      notification certificate-expiration {
        if-feature "certificate-expiration-notification";
        description
          "A notification indicating that the configured certificate
           is either about to expire or has already expired.  When to
           send notifications is an implementation specific decision,
           but it is RECOMMENDED that a notification be sent once a
           month for 3 months, then once a week for four weeks, and
           then once a day thereafter until the issue is resolved.

           If the certificate's Issuer maintains a Certificate
           Revocation List (CRL), the expiration notification MAY
           be sent if the CRL is about to expire.";
        leaf expiration-date {
          type yang:date-and-time;
          mandatory true;
          description
            "Identifies the expiration date on the certificate.";
        }
      }
    }

    grouping trust-anchor-cert-grouping {
      description
        "A trust anchor certificate, and a notification for when
         it is about to (or already has) expire.";
      leaf cert-data {
        nacm:default-deny-all;
        type trust-anchor-cert-cms;
        description
          "The binary certificate data for this certificate.";
      }
      uses certificate-expiration-grouping;
    }

    grouping end-entity-cert-grouping {
      description
        "An end entity certificate, and a notification for when
         it is about to (or already has) expire.  Implementations
```

```
          SHOULD assert that, where used, the end entity certificate
          contains the expected public key.";
        leaf cert-data {
          nacm:default-deny-all;
          type end-entity-cert-cms;
          description
            "The binary certificate data for this certificate.";
        }
        uses certificate-expiration-grouping;
      }



      grouping generate-csr-grouping {
        description
          "Defines the 'generate-csr' action.";
        action generate-csr {
          if-feature "csr-generation";
          nacm:default-deny-all;
          description
            "Generates a certificate signing request structure for
             the associated asymmetric key using the passed subject
             and attribute values.

             This action statement is only available when the
             associated 'public-key-format' node's value is
             'subject-public-key-info-format'.";
          input {
            leaf csr-format {
              type identityref {
                base csr-format;
              }
              mandatory true;
              description
                "Specifies the format for the returned certificate.";
            }
            leaf csr-info {
              type csr-info;
              mandatory true;
              description
                "A CertificationRequestInfo structure, as defined in
                 RFC 2986.

                 Enables the client to provide a fully-populated
                 CertificationRequestInfo structure that the server
                 only needs to sign in order to generate the complete
                 'CertificationRequest' structure to return in the
                 'output'.
```

```
               The 'AlgorithmIdentifier' field contained inside
               the 'SubjectPublicKeyInfo' field MUST be one known
               to be supported by the device.";
            reference
              "RFC 2986:
                 PKCS #10: Certification Request Syntax Specification
               RFC AAAA:
                 YANG Data Types and Groupings for Cryptography";
          }
        }
        output {
          choice csr-type {
            mandatory true;
            description
              "A choice amongst certificate signing request formats.
               Additional formats MAY be augmented into this 'choice'
               statement by future efforts.";
            case p10-csr {
              leaf p10-csr {
                type p10-csr;
                description
                  "A CertificationRequest, as defined in RFC 2986.";
              }
              description
                "A CertificationRequest, as defined in RFC 2986.";
              reference
                "RFC 2986:
                   PKCS #10: Certification Request Syntax Specification
                 RFC AAAA:
                   YANG Data Types and Groupings for Cryptography";
            }
          }
        }
      }
    } // generate-csr-grouping

    grouping asymmetric-key-pair-with-cert-grouping {
      description
        "A private/public key pair and an associated certificate.
         Implementations MUST assert that the certificate contains
         the matching public key.";
      uses asymmetric-key-pair-grouping;
      uses end-entity-cert-grouping;
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-cert-grouping

    grouping asymmetric-key-pair-with-certs-grouping {
      description
```

```
        "A private/public key pair and a list of associated
         certificates.  Implementations MUST assert that
         certificates contain the matching public key.";
      uses asymmetric-key-pair-grouping;
      container certificates {
        nacm:default-deny-write;
        description
          "Certificates associated with this asymmetric key.";
        list certificate {
          key "name";
          description
            "A certificate for this asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the certificate.";
          }
          uses end-entity-cert-grouping {
            refine "cert-data" {
              mandatory true;
            }
          }
        }
      }
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-certs-grouping

  }

  <CODE ENDS>
```

3. Security Considerations

3.1. No Support for CRMF

This document uses PKCS #10 [RFC2986] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [RFC4211] was considered, but it was unclear if there was market demand for it. If it is desired to support CRMF in the future, a backwards compatible solution can be defined at that time.

3.2. No Support for Key Generation

Early revisions of this document included "rpc" statements for generating symmetric and asymmetric keys. These statements were removed due to an inability to obtain consensus for how to generically identify the key-algorithm to use. Hence, the solution presented in this document only supports keys to be configured via an external client.

Separate protocol-specific modules can present protocol-specific key-generating RPCs (e.g., the "generate-public-key" RPC in [I-D.ietf-netconf-ssh-client-server] and [I-D.ietf-netconf-tls-client-server]).

3.3. Unconstrained Public Key Usage

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

The "asymmetric-key-pair-grouping" grouping uses the aforementioned "public-key-grouping" grouping, and carries the same traits.

The "asymmetric-key-pair-with-cert-grouping" grouping uses the aforementioned "asymmetric-key-pair-grouping" grouping, whereby associated certificates MUST constrain the usage of the public key according to local policy.

3.4. Unconstrained Private Key Usage

This module defines the "asymmetric-key-pair-grouping" grouping, which enables the configuration of private keys without constraints on their usage, e.g., what operations the key is allowed to be used for (e.g., signature, decryption, both).

The "asymmetric-key-pair-with-cert-grouping" uses the aforementioned "asymmetric-key-pair-grouping" grouping, whereby configured certificates (e.g., identity certificates) may constrain the use of the public key according to local policy.

3.5. Cleartext Passwords and Keys

The module contained within this document enables, only when specific "feature" statements are enabled, for the cleartext value of passwords and keys to be stored in the configuration database. Storing cleartext values for passwords and keys is NOT RECOMMENDED.

3.6. Encrypting Passwords and Keys

The module contained within this document enables cleartext passwords and keys to be encrypted via another key, either symmetric or asymmetric. Both formats use a CMS structure (EncryptedData and EnvelopedData respectively), which allows any encryption algorithm to be used.

To securely encrypt a password or key with a symmetric key, a proper block cipher mode such as an AEAD or CBC MUST be used. This ensures that a random IV is part of the input, which guarantees that the output for encrypting the same password or key still produces a different unpredictable ciphertext. This avoids leaking that some encrypted keys or passwords are the same and makes it much harder to pre-generate rainbow tables to brute force attack weak passwords. The ECB block cipher mode MUST NOT be used.

3.7. Deletion of Cleartext Key Values

This module defines storage for cleartext key values that SHOULD be zeroized when deleted, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

The cleartext key values are the "cleartext-symmetric-key" node defined in the "symmetric-key-grouping" grouping (Section 2.1.4.3) and the "cleartext-private-key" node defined in the "asymmetric-key-pair-grouping" grouping ("Section 2.1.4.6).

3.8. Considerations for the "ietf-crypto-types" YANG Module

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

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

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

Since the module in this document only defines groupings, these considerations are primarily for the designers of other modules that use these groupings.

Some of the readable data nodes defined in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. The following subtrees and data nodes have particular sensitivity/vulnerability:

* The "cleartext-password" node:

  The "cleartext-password" node defined in the "password-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

* The "cleartext-symmetric-key" node:

  The "cleartext-symmetric-key" node defined in the "symmetric-key-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

* The "cleartext-private-key" node:

  The "cleartext-private-key" node defined in the "asymmetric-key-pair-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied.

* The "cert-data" node:

  The "cert-data" node, defined in both the "trust-anchor-cert-grouping" and "end-entity-cert-grouping" groupings, is additionally sensitive to read operations, as certificates may provide insight into which other resources/applications/servers this particular server communicates with, as well as potentially divulge personally identifying information (e.g., end-entity certificates). For this reason, the NACM extension "default-deny-all" has been applied.

All the writable data nodes defined by all the groupings defined in this module may be considered sensitive or vulnerable in some network environments. For instance, even the modification of a public key or a certificate can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been applied to all the data nodes defined in the module.

Some of the operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

* generate-certificate-signing-request:

  This "action" statement SHOULD only be executed by authorized users. For this reason, the NACM extension "default-deny-all" has been applied. Note that NACM uses "default-deny-all" to protect "RPC" and "action" statements; it does not define, e.g., an extension called "default-deny-execute".

  For this action, it is RECOMMENDED that implementations assert channel binding [RFC5056], so as to ensure that the application layer that sent the request is the same as the device authenticated when the secure transport layer was established.

4. IANA Considerations

4.1. 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-crypto-types
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

4.2. The "YANG Module Names" Registry

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

```
name:         ietf-crypto-types
namespace:    urn:ietf:params:xml:ns:yang:ietf-crypto-types
prefix:       ct
reference:    RFC AAAA
```

5. References

5.1. Normative References

[ITU.X680.2021]
International Telecommunication Union, "Information technology - Abstract Syntax Notation One (ASN.1):

Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680-202102-I>.

[ITU.X690.2021] International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690-202102-I>.

[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>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC6031] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Symmetric Key Package Content Type", RFC 6031, DOI 10.17487/RFC6031, December 2010, <https://www.rfc-editor.org/info/rfc6031>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7093] Turner, S., Kent, S., and J. Manger, "Additional Methods for Generating Key Identifiers Values", RFC 7093, DOI 10.17487/RFC7093, December 2013, <https://www.rfc-editor.org/info/rfc7093>.

[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>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[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>.

5.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

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

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <https://www.rfc-editor.org/info/rfc5056>.

[RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", RFC 5915, DOI 10.17487/RFC5915, June 2010, <https://www.rfc-editor.org/info/rfc5915>.

[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>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[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>.

[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>.

Appendix A. Change Log

A.1. I-D to 00

* Removed groupings and notifications.
* Added typedefs for identityrefs.
* Added typedefs for other RFC 5280 structures.
* Added typedefs for other RFC 5652 structures.
* Added convenience typedefs for RFC 4253, RFC 5280, and RFC 5652.

A.2. 00 to 01

* Moved groupings from the draft-ietf-netconf-keystore here.

A.3. 01 to 02

* Removed unwanted "mandatory" and "must" statements.
* Added many new crypto algorithms (thanks Haiguang!)
* Clarified in asymmetric-key-pair-with-certs-grouping, in certificates/certificate/name/description, that if the name MUST NOT match the name of a certificate that exists independently in <operational>, enabling certs installed by the manufacturer (e.g., an IDevID).

A.4. 02 to 03

* renamed base identity 'asymmetric-key-encryption-algorithm' to 'asymmetric-key-algorithm'.
* added new 'asymmetric-key-algorithm' identities for secp192r1, secp224r1, secp256r1, secp384r1, and secp521r1.
* removed 'mac-algorithm' identities for mac-aes-128-ccm, mac-aes-192-ccm, mac-aes-256-ccm, mac-aes-128-gcm, mac-aes-192-gcm, mac-aes-256-gcm, and mac-chacha20-poly1305.

* for all -cbc and -ctr identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-algorithm'.

* for all -ccm and -gcm identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-and-mac-algorithm' and renamed the identity to remove the "enc-" prefix.

* for all the 'signature-algorithm' based identities, renamed from 'rsa-*' to 'rsassa-*'.

* removed all of the "x509v3-" prefixed 'signature-algorithm' based identities.

* added 'key-exchange-algorithm' based identities for 'rsaes-oaep' and 'rsaes-pkcs1-v1_5'.

* renamed typedef 'symmetric-key-encryption-algorithm-ref' to 'symmetric-key-algorithm-ref'.

* renamed typedef 'asymmetric-key-encryption-algorithm-ref' to 'asymmetric-key-algorithm-ref'.

* added typedef 'encryption-and-mac-algorithm-ref'.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.5. 03 to 04

* ran YANG module through formatter.

A.6. 04 to 05

* fixed broken symlink causing reformatted YANG module to not show.

A.7. 05 to 06

* Added NACM annotations.

* Updated Security Considerations section.

* Added 'asymmetric-key-pair-with-cert-grouping' grouping.

* Removed text from 'permanently-hidden' enum regarding such keys not being backed up or restored.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Added an explanation to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements as for why the nodes are not mandatory (e.g., because they may exist only in <operational>.

* Added 'must' expressions to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements ensuring sibling nodes are either all exist or do not all exist.

* Added an explanation to the 'permanently-hidden' that the value cannot be configured directly by clients and servers MUST fail any attempt to do so.

* Added 'trust-anchor-certs-grouping' and 'end-entity-certs-grouping' (the plural form of existing groupings).

* Now states that keys created in <operational> by the *-hidden-key actions are bound to the lifetime of the parent 'config true' node, and that subsequent invocations of either action results in a failure.

A.8. 06 to 07

* Added clarifications that implementations SHOULD assert that configured certificates contain the matching public key.

* Replaced the 'generate-hidden-key' and 'install-hidden-key' actions with special 'crypt-hash' -like input/output values.

A.9. 07 to 08

* Removed the 'generate-key and 'hidden-key' features.

* Added grouping symmetric-key-grouping

* Modified 'asymmetric-key-pair-grouping' to have a 'choice' statement for the keystone module to augment into, as well as replacing the 'union' with leafs (having different NACM settings.

A.10. 08 to 09

* Converting algorithm from identities to enumerations.

A.11. 09 to 10

* All the below changes are to the algorithm enumerations defined in ietf-crypto-types.

* Add in support for key exchange over x.25519 and x.448 based on RFC 8418.

* Add in SHAKE-128, SHAKE-224, SHAKE-256, SHAKE-384 and SHAKE 512

* Revise/add in enum of signature algorithm for x25519 and x448

* Add in des3-cbc-sha1 for IPSec

* Add in sha1-des3-kd for IPSec

* Add in definit for rc4-hmac and rc4-hmac-exp. These two algorithms have been deprecated in RFC 8429. But some existing draft in i2nsf may still want to use them.

* Add x25519 and x448 curve for asymmetric algorithms

* Add signature algorithms ed25519, ed25519-cts, ed25519ph

* add signature algorithms ed448, ed448ph

* Add in rsa-sha2-256 and rsa-sha2-512 for SSH protocols (rfc8332)

A.12. 10 to 11

* Added a "key-format" identity.

* Added symmetric keys to the example in Section 2.2.

A.13. 11 to 12

* Removed all non-essential (to NC/RC) algorithm types.

* Moved remaining algorithm types each into its own module.

* Added a 'config false' "algorithms-supported" list to each of the algorithm-type modules.

A.14. 12 to 13

* Added the four features: "[encrypted-]one-[a]symmetric-key-format", each protecting a 'key-format' identity of the same name.

* Added 'must' expressions asserting that the 'key-format' leaf exists whenever a non-hidden key is specified.

* Improved the 'description' statements and added 'reference' statements for the 'key-format' identities.

* Added a questionable forward reference to "encrypted-*" leafs in a couple 'when' expressions.

* Did NOT move "config false" alg-supported lists to SSH/TLS drafts.

A.15. 13 to 14

* Resolved the "FIXME: forward ref" issue by modulating 'must', 'when', and 'mandatory' expressions.

* Moved the 'generatesymmetric-key' and 'generate-asymmetric-key' actions from ietf-keystore to ietf-crypto-types, now as RPCs.

* Cleaned up various description statements and removed lingering FIXMEs.

* Converted the "iana-<alg-type>-algs" YANG modules to IANA registries with instructions for how to generate modules from the registries, whenever they may be updated.

A.16. 14 to 15

* Removed the IANA-maintained registries for symmetric, asymmetric, and hash algorithms.

* Removed the "generate-symmetric-key" and "generate-asymmetric-key" RPCs.

* Removed the "algorithm" node in the various symmetric and asymmetric key groupings.

* Added 'typedef csr' and 'feature certificate-signing-request-generation'.

* Refined a usage of "end-entity-cert-grouping" to make the "cert" node mandatory true.

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

A.17. 15 to 16

* Updated draft title (refer to "Groupings" too).

* Removed 'end-entity-certs-grouping' as it wasn't being used anywhere.

* Removed 'trust-anchor-certs-grouping' as it was no longer being used after modifying 'inline-or-truststore-certs-grouping' to use lists (not leaf-lists).

* Renamed "cert" to "cert-data" in trust-anchor-cert-grouping.

* Added "csr-info" typedef, to complement the existing "csr" typedef.

* Added "ocsp-request" and "ocsp-response" typedefs, to complement the existing "crl" typedef.

* Added "encrypted" cases to both symmetric-key-grouping and asymmetric-key-pair-grouping (Moved from Keystore draft).

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

* Updated the Security Considerations section.

A.18. 16 to 17

* [Re]-added a "Strength of Keys Configured" Security Consideration

* Prefixed "cleartext-" in the "key" and "private-key" node names.

A.19. 17 to 18

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

* Added "password-grouping", discussed during the IETF 108 session.

A.20. 18 to 19

* Added a "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.

* Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.

* Changed the encryption strategy, after conferring with Russ Housley.

* Added a "password-grouping" example to the "crypto-types-usage" example.

* Added an "Encrypting Passwords" section to Security Consideration.

* Addressed other comments raised by YANG Doctor.

A.21. 19 to 20

* Nits found via YANG Doctors reviews.

* Aligned modules with ‘pyang -f‘ formatting.

A.22. 20 to 21

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

* Accommodated SecDir review by Valery Smyslov.

A.23. 21 to 22

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

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

* added ’hidden-keys’ feature.

A.24. 22 to 23

* Fixed an example to reference correct key.

* Fixed an example to not have line-returns around the encoding for a binary value.

A.25. 23 to 24

* Added mandatory leaf "csr-format" to action "generate-csr".

* s/certificate-signing-request/csr/g in the YANG module.

A.26. 24 to 25

* Updated per Shepherd reviews impacting the suite of drafts.

A.27. 25 to 26

* Updated per Shepherd reviews impacting the suite of drafts.

A.28. 26 to 27

* Updated per Tom Petch and AD reviews.

* Renamed numerous "feature" statements and some "grouping" statements (in YANG)

* Added "csr-format" and "p10-csr-format" identities to doc (they were already in YANG)

* Clarified that the 'rsa-private-key-format' and 'ec-private-key-format' formats must be encoded using DER

* Added 'if-feature cleartext-passwords' statement to 'case cleartext-password' in grouping 'password-grouping'.

* Added 'if-feature cleartext-keys' statement to 'case cleartext-key' in grouping 'symmetric-key-grouping'.

* Added 'if-feature cleartext-cleartext-private-keys' statement to 'case cleartext-private-key' in grouping 'asymmetric-key-grouping'.

* Updated Section titles.

* Clarified Security Considerations about the "generate-public-key" RPCs.

A.29. 27 to 28

* Mostly addresses AD review comments.

* Also addresses on-list comment regarding public-keys being "mandatory true."

* Added note to Editor to fix line foldings.

* Factored 'private-key-grouping' from 'asymmetric-key-pair-grouping'.

* Made public-key in 'asymmetric-key-pair-grouping' be "mandatory false".

* Renamed 'encrypted-by-choice-grouping' to 'encrypted-by-grouping'.

A.30. 28 to 29

* Addresses Gen-ART review by Dale Worley.

* Addresses review by Tom Petch.

A.31. 29 to 30

* Addresses 1st-round of IESG reviews.

A.32. 30 to 32

* Addresses issues found in OpsDir of the ssh-client-server draft.

* Removed "Strength of Keys Conveyed" section.

* Renamed Security Considerations section s/Template for/ Considerations for/

* Improved Security Consideration for 'cert-data' node.

A.33. 32 to 34

* Nothing changed. Only bumped for automation...

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Balázs Kovács, Carsten Bormann, Dale Worley, Eric Voit, Éric Vyncke, Francesca Palombini, Jürgen Schönwälder, Lars Eggert, Liang Xia, Martin Björklund, Mahesh Jethanandani, Murray Kucherawy, Nick Hancock, Orie Steele, Paul Wouters, Rich Salz, Rifaat Shekh-Yusef, Rob Wilton, Roman Danyliw, Russ Housley, Sandra Murphy, Tom Petch, Valery Smyslov, Wang Haiguang, Warren Kumari, and Zaheduzzaman Sarker.

Author's Address

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

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 January 2021

K. Watsen
Watsen Networks

8 July 2020

YANG Groupings for HTTP Clients and HTTP Servers
draft-ietf-netconf-http-client-server-04

Abstract

This document defines two YANG modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based protocols (not for complete web servers or browsers).

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 (note: not all may be present):

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

* "BBBB" --> the assigned RFC value for draft-ietf-netconf-trust-anchors

* "CCCC" --> the assigned RFC value for draft-ietf-netconf-keystore

* "DDDD" --> the assigned RFC value for draft-ietf-netconf-tcp-client-server

* "EEEE" --> the assigned RFC value for draft-ietf-netconf-ssh-client-server

* "FFFF" --> the assigned RFC value for draft-ietf-netconf-tls-client-server

* "GGGG" --> the assigned RFC value for this draft

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

* "2020-07-08" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on 9 January 2021.

Copyright Notice

Table of Contents

1. Introduction

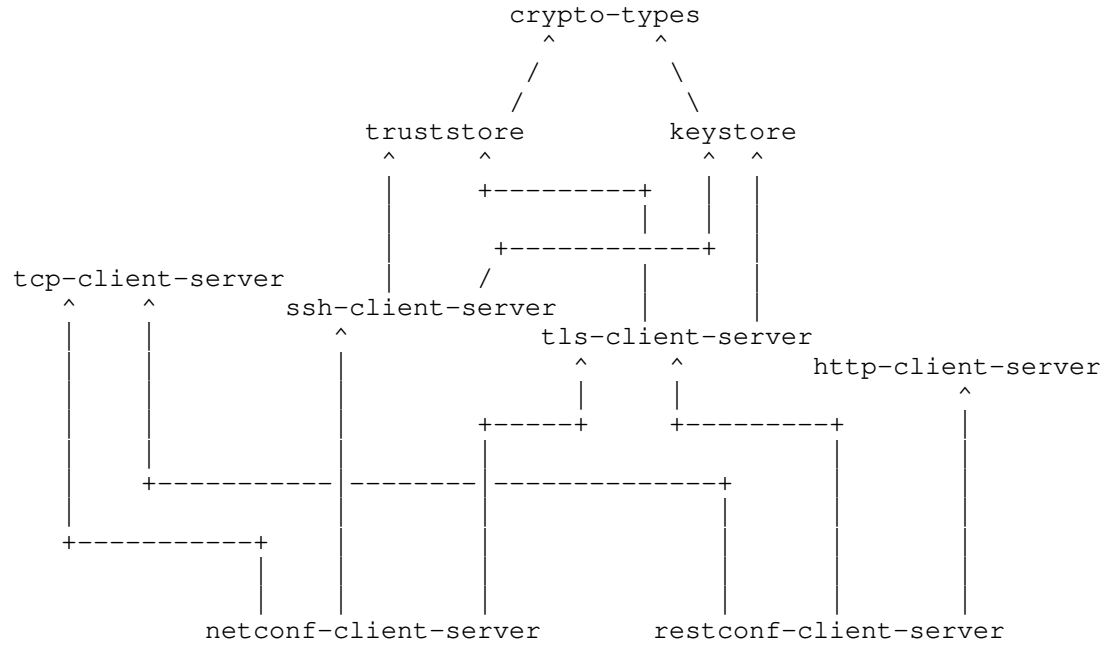
This document defines two YANG 1.1 [RFC7950] modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based protocols (not for complete web servers or browsers).

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

2. The "ietf-http-client" Module

2.1. Data Model Overview

2.1.1. Features

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

```
Features:
  +-- proxy-connect
  +-- basic-auth
  +-- tcp-supported
  +-- tls-supported
```

2.1.2. Groupings

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

```
Groupings:
  +-- http-client-identity-grouping
  +-- http-client-grouping
  +-- http-client-stack-grouping
```

Each of these groupings are presented in the following subsections.

2.1.2.1. The "http-client-identity-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-client-identity-grouping" grouping:

```
  grouping http-client-identity-grouping
    +-- client-identity!
       +-- (auth-type)
          +--:(basic)
             +-- basic {basic-auth}?
                +-- user-id     string
                +-- password    string
```

Comments:

* This grouping exists because it is used three times by the "http-client-grouping" discussed in Section 2.1.2.2.

* The "client-identity" node is a "presence" container so that its descendent "choice" node's "mandatory true" doesn't imply that a client identity must be configured, as a client identity may be configured at protocol layers.

* The "basic" authentication scheme is the only scheme defined by this module, albeit it must be enabled via the "basic-auth" feature (see Section 2.1.1).

* Other authentication schemes MAY be augmented in as needed by the application.

2.1.2.2. The "http-client-grouping" Grouping

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

```
  grouping http-client-grouping
    +---u http-client-identity-grouping
    +-- proxy-connect! {proxy-connect}?
       +-- (proxy-type)
          +--:(http)
          |  +-- http-proxy
          |     +-- tcp-client-parameters
          |     |  +---u tcpc:tcp-client-grouping
          |     +-- http-client-parameters
          |        +---u http-client-identity-grouping
          +--:(https)
             +-- https-proxy
                +-- tcp-client-parameters
                |  +---u tcpc:tcp-client-grouping
                +-- tls-client-parameters
                |  +---u tlsc:tls-client-grouping
                +-- http-client-parameters
                   +---u http-client-identity-grouping
```

Comments:

* The "http-client-grouping" defines the configuration for just "HTTP" part of a protocol stack. It does not, for instance, define any configuration for the "TCP" or "TLS" protocol layers (for that, see Section 2.1.2.3).

* Beyond configuring the client's identity, via the "http-client-identity-grouping" grouping discussed in Section 2.1.2.1, this grouping defines support for HTTP-proxies, albeit it must be enabled via a "feature" statement.

* The "proxy-connect" node is a "presence" container so that its descendent "choice" node's "mandatory true" doesn't imply that a proxy connection must be configured, assuming the server supports the "proxy-connect" feature.

* For the referenced grouping statement(s):

  - The "http-client-identity-grouping" grouping is discussed in Section 2.1.2.1.
  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].

2.1.2.3. The "http-client-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-client-stack-grouping" grouping:

```
  grouping http-client-stack-grouping
    +-- (transport)
       +--:(tcp) {tcp-supported}?
       |  +-- tcp
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- http-client-parameters
       |        +---u http-client-grouping
       +--:(tls) {tls-supported}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
                +---u http-client-grouping
```

Comments:

* The "http-client-stack-grouping" is a convenience grouping for downstream modules. It defines both the "HTTP" and "HTTPS" protocol stacks, with each option enabled by a "feature" statement for application control.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 in this document.

2.1.3. Protocol-accessible Nodes

The "ietf-http-client" module does not contain any protocol-accessible nodes.

2.2. Example Usage

This section presents two examples showing the http-client-grouping populated with some data.

The following example illustrates an HTTP client connecting directly to an HTTP server.

```
<http-client xmlns="urn:ietf:params:xml:ns:yang:ietf-http-client">
  <client-identity>
    <basic>
      <user-id>bob</user-id>
      <password>secret</password>
    </basic>
  </client-identity>
</http-client>
```

The following example illustrates the same client connecting through an HTTP proxy. This example is consistent with examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

   <http-client xmlns="urn:ietf:params:xml:ns:yang:ietf-http-client">
     <client-identity>
       <basic>
         <user-id>bob</user-id>
         <password>secret</password>
       </basic>
     </client-identity>
     <proxy-connect>
       <https-proxy>
         <tcp-client-parameters>
           <remote-address>corp-fw2.example.com</remote-address>
           <keepalives>
             <idle-time>15</idle-time>
             <max-probes>3</max-probes>
             <probe-interval>30</probe-interval>
           </keepalives>
         </tcp-client-parameters>
         <tls-client-parameters>
           <client-identity>
             <certificate>
               <keystore-reference>
                 <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                 <certificate>ex-rsa-cert</certificate>
               </keystore-reference>
             </certificate>
           </client-identity>
           <server-authentication>
             <ca-certs>
               <truststore-reference>trusted-server-ca-certs</truststor\
   e-reference>
             </ca-certs>
             <ee-certs>
               <truststore-reference>trusted-server-ee-certs</truststor\
   e-reference>
             </ee-certs>
           </server-authentication>
         </tls-client-parameters>
         <http-client-parameters>
           <client-identity>
             <basic>
               <user-id>local-app-1</user-id>
               <password>secret</password>
             </basic>
           </client-identity>
         </http-client-parameters>
       </https-proxy>
```

```
    </proxy-connect>
  </http-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6991].

<CODE BEGINS> file "ietf-http-client@2020-07-08.yang"

```
module ietf-http-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-http-client";
  prefix httpc;

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

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC AAAA: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC BBBB: YANG Groupings for TLS Clients and TLS Servers";
  }

  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module defines reusable groupings for HTTP clients that
     can be used as a basis for specific HTTP client instances.

     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 GGGG
        (https://www.rfc-editor.org/info/rfcGGGG); see the RFC
        itself for full legal notices.

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

    revision 2020-07-08 {
      description
        "Initial version";
      reference
        "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
    }

    // Features

    feature proxy-connect {
      description
        "Proxy connection configuration is configurable for
         HTTP clients on the server implementing this feature.";
    }

    feature basic-auth {
      description
        "The 'basic-auth' feature indicates that the client
         may be configured to use the 'basic' HTTP authentication
         scheme.";
      reference
        "RFC 7617: The 'Basic' HTTP Authentication Scheme";
    }

    feature tcp-supported {
      description
        "Indicates that the server supports HTTP/TCP.";
    }

    feature tls-supported {
      description
```

```
        "Indicates that the server supports HTTP/TLS.";
    }

    // Groupings

    grouping http-client-identity-grouping {
      description
        "A grouping to provide HTTP credentials used by the
         client to authenticate itself to the HTTP server.";
      container client-identity {
        nacm:default-deny-write;
        presence
          "Indicates that HTTP-level client authentication
           is sent.  Present so that the 'choice' node's
           mandatory true doesn't imply that a client
           identity must be configured.";
        description
          "The identity the HTTP client should use when
           authenticating itself to the HTTP server.";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst available authentication types.";
          case basic {
            container basic {
              if-feature "basic-auth";
              leaf user-id {
                type string;
                mandatory true;
                description
                  "The user-id for the authenticating client.";
              }
              leaf password {
                nacm:default-deny-all;
                type string;
                mandatory true;
                description
                  "The password for the authenticating client.";
              }
              description
                "The 'basic' HTTP scheme credentials.";
              reference
                "RFC 7617: The 'Basic' HTTP Authentication Scheme";
            }
          }
        }
      }
```

```
    } // grouping http-client-identity-grouping

    grouping http-client-grouping {
      description
        "A reusable grouping for configuring a HTTP client.

         This grouping is expected to be used in conjunction with
         other configurations providing, e.g., the hostname or IP
         address and port number the client initiates connections
         to.

         Note that this grouping uses fairly typical descendent
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'http-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.

         FIXME: it is assumed that the application can construct any
         necessary HTTP path, e.g., via a YANG 'rpc' definition...ok?";

      uses http-client-identity-grouping;

      container proxy-connect {
        nacm:default-deny-write;
        if-feature "proxy-connect";
        presence
          "Indicates that the HTTP-client is to connect thru an
           HTTP-level proxy server.  Present so that the 'choice'
           node's mandatory true doesn't imply that a proxy
           connection must be configured.";
        choice proxy-type {
          mandatory true;
          description
            "Choice amongst proxy server types.";
          case http {
            container http-proxy {
              /* FIXME: THIS ISN'T NEEDED...because in a 'case'
              presence
                "Indicates that a HTTP proxy (Web proxy) connection
                 is configured.";
              */
              description
                "Container for HTTP Proxy (Web Proxy) server
                 configuration parameters.";
              container tcp-client-parameters {
```

```
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses "tcpc:tcp-client-grouping";
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-client-identity-grouping;
              /* FIXME: is HTTP Proxy auth *required*
                refine client-identity/auth-type {
                  mandatory true;
                }
              */
            }
          }
        }
        case https {
          container https-proxy {
            /* FIXME: THIS ISN'T NEEDED...because in a 'case'
            presence
              "Indicates that a HTTPS proxy (Secure web proxy)
               connection is configured.";
            */
            /* FIXME: is HTTP Proxy auth *required*
            must
              "tls-client-parameters/client-identity
               or http-client-parameters/client-identity";
            */
            description
              "Container for HTTPS Proxy (Secure Web Proxy) server
               configuration parameters.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses "tcpc:tcp-client-grouping";
            }
            container tls-client-parameters {
              description
                "A wrapper around the TLS parameters to avoid
                 name collisions.";
              uses "tlsc:tls-client-grouping";
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
```

```
                   name collisions.";
                 uses http-client-identity-grouping;
               }
             }
           }
         }
         description
           "Proxy server settings.";
       }
     } // grouping http-client-grouping


     grouping http-client-stack-grouping {
       description
         "A grouping that defines common HTTP-based protocol stacks.";
       choice transport {
         mandatory true;
         description
           "Choice amongst various transports type.  TCP, with and
            without TLS are defined here, with 'feature' statements
            so that they may be disabled.  Other transports MAY be
            augmented in as 'case' statements by future efforts.";
         case tcp {
           if-feature tcp-supported;
           container tcp {
             description
               "Container for TCP-based HTTP protocols.";
             container tcp-client-parameters {
               description
                 "A wrapper around the TCP parameters to avoid
                  name collisions.";
               uses "tcpc:tcp-client-grouping";
             }
             container http-client-parameters {
               description
                 "A wrapper around the HTTP parameters to avoid
                  name collisions.";
               uses http-client-grouping;
             }
           }
         }
         case tls {
           if-feature tls-supported;
           container tls {
             description
               "Container for TLS-based HTTP protocols.";
             container tcp-client-parameters {
```

```
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses "tcpc:tcp-client-grouping";
            }
            container tls-client-parameters {
              description
                "A wrapper around the TLS parameters to avoid
                 name collisions.";
              uses "tlsc:tls-client-grouping";
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-client-grouping;
            }
          }
        }
      }
    }

  } // module ietf-http-client

  <CODE ENDS>
```

3. The "ietf-http-server" Module

3.1. Data Model Overview

3.1.1. Features

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

```
Features:
  +-- client-auth-config-supported
  +-- basic-auth
  +-- tcp-supported
  +-- tls-supported
```

3.1.2. Groupings

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

```
Groupings:
  +-- http-server-grouping
  +-- http-server-stack-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "http-server-grouping" Grouping

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

```
  grouping http-server-grouping
    +-- server-name?             string
    +-- client-authentication! {client-auth-config-supported}?
       +-- users
          +-- user* [user-id]
             +-- user-id?       string
             +-- (auth-type)?
                +--:(basic)
                   +-- basic {basic-auth}?
                      +-- user-id?    string
                      +-- password?   ianach:crypt-hash
```

Comments:

* The "http-server-grouping" defines the configuration for just "HTTP" part of a protocol stack. It does not, for instance, define any configuration for the "TCP" or "TLS" protocol layers (for that, see Section 3.1.2.2).

* The "server-name" node defines the HTTP server's name, as presented to HTTP clients.

* The "client-authentication" node, which must by enabled by a feature, defines a very simple user-database. Only the "basic" authentication scheme is supported, albiet it must be enabled by a "feature". Other authentication schemes MAY be augmented in.

3.1.2.2. The "http-server-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-server-stack-grouping" grouping:

```
  grouping http-server-stack-grouping
    +-- (transport)
       +--:(tcp) {tcp-supported}?
       |  +-- tcp
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-server-parameters
       |        +---u http-server-grouping
       +--:(tls) {tls-supported}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
                +---u http-server-grouping
```

Comments:

* The "http-server-stack-grouping" is a convenience grouping for downstream modules. It defines both the "HTTP" and "HTTPS" protocol stacks, with each option enabled by a "feature" statement for application control.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-http-server" module does not contain any protocol-accessible nodes.

3.2. Example Usage

This section presents an example showing the http-server-grouping populated with some data.

```
<http-server xmlns="urn:ietf:params:xml:ns:yang:ietf-http-server">
  <server-name>foo.example.com</server-name>
</http-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6991].

<CODE BEGINS> file "ietf-http-server@2020-07-08.yang"

```
module ietf-http-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-http-server";
  prefix https;

  import iana-crypt-hash {
    prefix ianach;
    reference
      "RFC 7317: A YANG Data Model for System Management";
  }

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

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC AAAA: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-server {
    prefix tlss;
    reference
      "RFC BBBB: YANG Groupings for TLS Clients and TLS Servers";
  }

  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module defines reusable groupings for HTTP servers that
     can be used as a basis for specific HTTP server instances.

     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 GGGG
        (https://www.rfc-editor.org/info/rfcGGGG); see the RFC
        itself for full legal notices.

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

      revision 2020-07-08 {
        description
          "Initial version";
        reference
          "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
      }

      // Features

      feature client-auth-config-supported {
        description
          "Indicates that the configuration for how to authenticate
           clients can be configured herein, as opposed to in an
           application specific location.  That is, to support the
           consuming data models that prefer to place client
           authentication with client definitions, rather then
           in a data model principally concerned with configuring
           the transport.";
      }

      feature basic-auth {
        description
          "The 'basic-auth' feature indicates that the server
           may be configured authenticate users using the 'basic'
           HTTP authentication scheme.";
        reference
          "RFC 7617: The 'Basic' HTTP Authentication Scheme";
      }
```

```
     feature tcp-supported {
       description
         "Indicates that the server supports HTTP/TCP.";
     }

     feature tls-supported {
       description
         "Indicates that the server supports HTTP/TLS.";
     }

     // Groupings

     grouping http-server-grouping {
       description
         "A reusable grouping for configuring an HTTP server.

          Note that this grouping uses fairly typical descendent
          node names such that a stack of 'uses' statements will
          have name conflicts.  It is intended that the consuming
          data model will resolve the issue (e.g., by wrapping
          the 'uses' statement in a container called
          'http-server-parameters').  This model purposely does
          not do this itself so as to provide maximum flexibility
          to consuming models.";

       leaf server-name {
         nacm:default-deny-write;
         type string;
         description
           "The value of the 'Server' header field.  If not set, then
            underlying software's default value is used.  Set to the
            empty string to disable.";
       }

       container client-authentication {
         if-feature "client-auth-config-supported";
         nacm:default-deny-write;
         presence
           "Indicates that HTTP based client authentication is
            supported (i.e., the server will request that the
            HTTP client send authenticate when needed).  This
            is needed as some HTTP-based protocols may only
            support, e.g., TLS-level client authentication.";
         description
           "Specifies how the HTTP server can authenticate HTTP
            clients.";
         container users {
           description
```

```
            "A list of locally configured users.";
          list user {
            key user-id;
            description
              "The list of local users configured on this device.";
            leaf user-id {
              type string;
              description
                "The user-id for the authenticating client.";
            }
            choice auth-type {
              description
                "The authentication type.";
              container basic {
                if-feature "basic-auth";
                leaf user-id {
                  type string;
                  description
                    "The user-id for the authenticating client.";
                }
                leaf password {
                  nacm:default-deny-write;
                  type ianach:crypt-hash;
                  description
                    "The password for the authenticating client.";
                }
                description
                  "The 'basic' HTTP scheme credentials.";
                reference
                  "RFC 7617:
                    The 'Basic' HTTP Authentication Scheme";
              }
            }
          }
        }
      } // container client-authentication
    } // grouping http-server-grouping


    grouping http-server-stack-grouping {
      description
        "A grouping that defines common HTTP-based protocol stacks.";
      choice transport {
        mandatory true;
        description
          "Choice amongst various transports type.  TCP, with and
           without TLS are defined here, with 'feature' statements
           so that they may be disabled.  Other transports MAY be
```

```
           augmented in as 'case' statements by future efforts.";
        case tcp {
          if-feature tcp-supported;
          container tcp {
            description
              "Container for TCP-based HTTP protocols.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses "tcps:tcp-server-grouping";
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-server-grouping;
            }
          }
        }
        case tls {
          if-feature tls-supported;
          container tls {
            description
              "Container for TLS-based HTTP protocols.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP parameters to avoid
                 name collisions.";
              uses "tcps:tcp-server-grouping";
            }
            container tls-server-parameters {
              description
                "A wrapper around the TLS parameters to avoid
                 name collisions.";
              uses "tlss:tls-server-grouping";
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP parameters to avoid
                 name collisions.";
              uses http-server-grouping;
            }
          }
        }
      }
    }
  }
```

<CODE ENDS>

4. Security Considerations

4.1. The "ietf-http-client" YANG Module

The "ietf-http-client" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "client-identity/basic/password" node:

  The cleartext "password" node defined in the "http-client-identity-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

  | Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

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

> Please be aware that this module uses groupings from the "ietf-tls-client" and "ietf-tls-server" modules defined in [I-D.ietf-netconf-tls-client-server]. All of the data nodes defined in these groupings have the NACM extension "default-deny-write" set, thus preventing unrestricted write-access to the data nodes defined in those groupings.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

4.2. The "ietf-http-server" YANG Module

The "ietf-http-server" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

> Please be aware that this module uses groupings from the "ietf-tls-client" and "ietf-tls-server" modules defined in [I-D.ietf-netconf-tls-client-server]. All of the data nodes defined in these groupings have the NACM extension "default-deny-write" set, thus preventing unrestricted write-access to the data nodes defined in those groupings.

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 two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-http-client
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-http-server
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 two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-http-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-http-client
prefix:       httpc
reference:    RFC XXXX

name:         ietf-http-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-http-server
prefix:       https
reference:    RFC XXXX
```

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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

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

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

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

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

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

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

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

[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>.

Appendix A. Change Log

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

A.1. 00 to 01

* Modified Abstract and Intro to be more accurate wrt intended applicability.
* In ietf-http-client, removed "protocol-version" and all auth schemes except "basic".
* In ietf-http-client, factored out "client-identity-grouping" for proxy connections.
* In ietf-http-server, removed "choice required-or-optional" and "choice local-or-external".
* In ietf-http-server, moved the basic auth under a "choice auth-type" limited by new "feature basic-auth".

A.2. 01 to 02

* Removed the unused "external-client-auth-supported" feature from ietf-http-server.

A.3. 02 to 03

* Removed "protocol-versions" from ietf-http-server based on HTTP WG feedback.
* Slightly restructured the "proxy-server" definition in ietf-http-client.
* Added http-client example show proxy server use.
* Added a "Note to Reviewers" note to first page.

A.4. 03 to 04

* Added a parent "container" to "client-identity-grouping" so that it could be better used by the proxy model.
* Added a "choice" to the proxy model enabling selection of proxy types.
* Added 'http-client-stack-grouping' and 'http-server-stack-grouping' convenience groupings.

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

* Updated the Security Considerations section.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Mark Nottingham, Ben Schwartz, and Willy Tarreau.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

YANG Groupings for HTTP Clients and HTTP Servers
draft-ietf-netconf-http-client-server-20

Abstract

This document presents two YANG modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based protocols (not for complete web servers or browsers). Support is provided for HTTP/1.1, HTTP/2, and HTTP/3.

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 (note: not all may be present):

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
* GGGG --> 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:

* 2024-03-16 --> the publication date of this draft

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

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

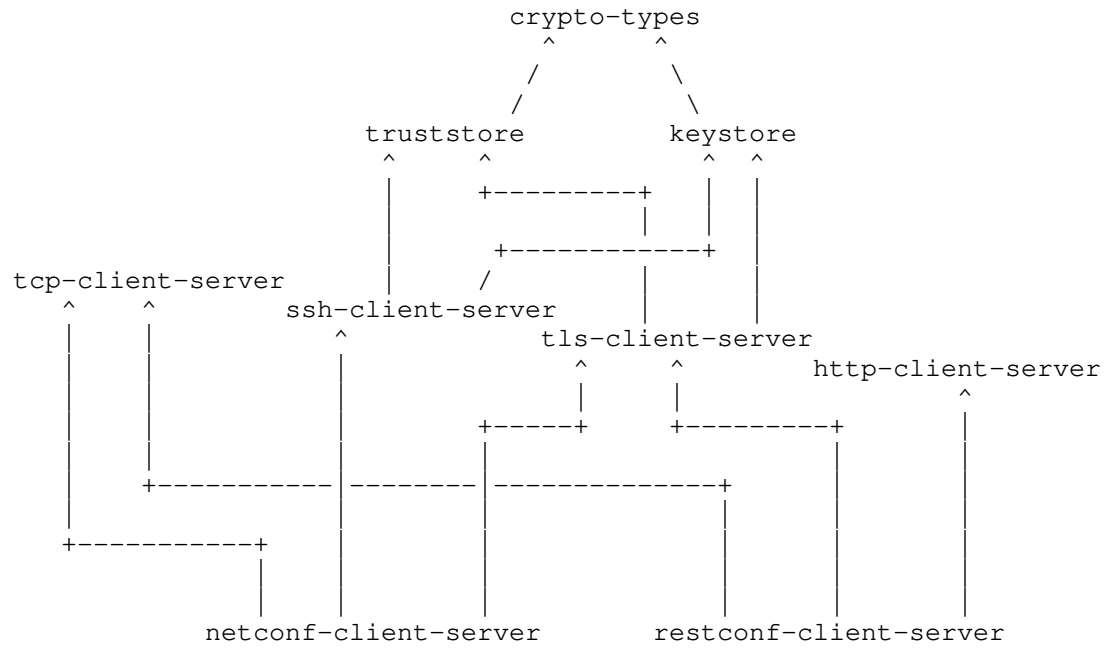
This document presents two YANG 1.1 [RFC7950] modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based protocols. Support is provided for HTTP/1.1, HTTP/2, and HTTP/3.

1.1. Relation to other RFCs

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

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



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

Table 1: Label 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, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational> (Section 5.3 of [RFC8342]), and <system> [I-D.ietf-netmod-system-config], if implemented.

2. The "ietf-http-client" Module

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

2.1. Data Model Overview

This section provides an overview of the "ietf-http-client" module in terms of its features and groupings.

2.1.1. Features

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

```
Features:
  +-- basic-auth
  +-- tcp-supported
  +-- tls-supported
  +-- proxy-connect
      +-- http-connect
      +-- https-connect
```

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

2.1.2. Groupings

The "ietf-http-client" module defines the following "grouping" statements:

* http-client-identity-grouping
* http-client-grouping
* http-client-stack-grouping

Each of these groupings are presented in the following subsections.

2.1.2.1. The "http-client-identity-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-client-identity-grouping" grouping:

```
  grouping http-client-identity-grouping:
    +-- client-identity!
       +-- (auth-type)
          +--:(basic)
             +-- basic {basic-auth}?
                +-- user-id                 string
                +---u ct:password-grouping
```

Comments:

* This grouping exists because it is used three times by the "http-client-grouping" discussed in Section 2.1.2.2.

* The "client-identity" node is a "presence" container so the mandatory descendant nodes do not imply that this node must be configured, as a client identity may be configured at protocol layers.

* The "basic" authentication scheme is the only scheme defined by this module, albeit it must be enabled via the "basic-auth" feature (see Section 2.1.1).

* Other authentication schemes MAY be augmented in as needed by the application.

2.1.2.2. The "http-client-grouping" Grouping

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

```
  grouping http-client-grouping:
    +---u http-client-identity-grouping
    +-- proxy-connect! {proxy-connect}?
       +-- (proxy-type)
          +--:(http) {http-connect-proxy}?
          |  +-- http-connect
          |     +-- tcp-client-parameters
          |     |  +---u tcpc:tcp-client-grouping
          |     +-- http-client-parameters
          |        +---u http-client-identity-grouping
          +--:(https) {https-connect-proxy}?
             +-- https-connect
                +-- tcp-client-parameters
                |  +---u tcpc:tcp-client-grouping
                +-- tls-client-parameters
                |  +---u tlsc:tls-client-grouping
                +-- http-client-parameters
                   +---u http-client-identity-grouping
```

Comments:

* The "http-client-grouping" primarily (not including the proxy configuration) defines the configuration for just the "HTTP" part of a protocol stack. It does not, for instance, define any configuration for the "TCP" or "TLS" protocol layers (for that, see Section 2.1.2.3).

* Beyond configuring the client's identity, via the "http-client-identity-grouping" grouping discussed in Section 2.1.2.1, this grouping defines support for HTTP-proxies, albeit it must be enabled via a "feature" statement.

* The "proxy-connect" node is a "presence" container so the mandatory descendant nodes do not imply that this node must be configured, assuming the server supports the "proxy-connect" feature.

* The "proxy-connect" node defines support for HTTP 1.1 and HTTP 2.0. Support for other protocol versions, e.g., HTTP/3, MAY be added by future work.

* For the referenced grouping statement(s):

  - The "http-client-identity-grouping" grouping is discussed in Section 2.1.2.1.
  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].

- The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].

2.1.2.3. The "http-client-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-client-stack-grouping" grouping:

```
  grouping http-client-stack-grouping:
    +-- (transport)
       +--:(tcp) {tcp-supported}?
       |  +-- tcp
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- http-client-parameters
       |        +---u http-client-grouping
       +--:(tls) {tls-supported}?
       |  +-- tls
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- tls-client-parameters
       |     |  +---u tlsc:tls-client-grouping
       |     +-- http-client-parameters
       |        +---u http-client-grouping
       +--:(quic) {quic-supported}?
          +-- quic
             +-- udp-client-parameters
             |  +---u udpc:udp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
                +---u http-client-grouping
```

Comments:

* The "http-client-stack-grouping" is a convenience grouping for consuming modules. It defines protocol stacks for HTTP/1.1, HTTP/2, and HTTP/3, with each option enabled by a "feature" statement for application control. Other protocols may be added by future work using the YANG "augment" statement.

* For the referenced grouping statement(s):

  - The "udp-client-grouping" grouping is discussed in Section 2 of [I-D.ietf-netconf-udp-client-server].
  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].

- The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
- The "http-client-grouping" grouping is discussed in Section 2.1.2.2 in this document.

2.1.3. Protocol-accessible Nodes

The "ietf-http-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not define any protocol-accessible nodes.

2.2. Example Usage

This section presents two examples showing the http-client-grouping populated with some data.

The following example illustrates the case where the HTTP client connects directly to an HTTP server. Note, the information identifying the remote server (e.g., its hostname) would be configured in the "tcp-client-grouping" (not shown).

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<http-client xmlns="urn:ietf:params:xml:ns:yang:ietf-http-client">
  <client-identity>
    <basic>
      <user-id>bob</user-id>
      <cleartext-password>example-secret</cleartext-password>
    </basic>
  </client-identity>
</http-client>
```

The following example illustrates the same client connecting through an HTTP proxy. This example is consistent with examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

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

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<http-client xmlns="urn:ietf:params:xml:ns:yang:ietf-http-client">
  <client-identity>
    <basic>
      <user-id>bob</user-id>
```

```
        <cleartext-password>example-secret</cleartext-password>
      </basic>
    </client-identity>
    <proxy-connect>
      <https-connect>
        <tcp-client-parameters>
          <remote-address>corp-fw2.example.com</remote-address>
          <keepalives>
            <idle-time>7200</idle-time>
            <max-probes>9</max-probes>
            <probe-interval>75</probe-interval>
          </keepalives>
        </tcp-client-parameters>
        <tls-client-parameters>
          <client-identity>
            <certificate>
              <central-keystore-reference>
                <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                <certificate>ex-rsa-cert</certificate>
              </central-keystore-reference>
            </certificate>
          </client-identity>
          <server-authentication>
            <ca-certs>
              <central-truststore-reference>trusted-server-ca-certs</c\
entral-truststore-reference>
            </ca-certs>
            <ee-certs>
              <central-truststore-reference>trusted-server-ee-certs</c\
entral-truststore-reference>
            </ee-certs>
          </server-authentication>
        </tls-client-parameters>
        <http-client-parameters>
          <client-identity>
            <basic>
              <user-id>local-app-1</user-id>
              <cleartext-password>example-secret</cleartext-password>
            </basic>
          </client-identity>
        </http-client-parameters>
      </https-connect>
    </proxy-connect>
  </http-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6991] [RFC7617], [RFC9110], [RFC9114], [I-D.ietf-netconf-crypto-types], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-tls-client-server], and [I-D.ietf-netconf-udp-client-server].

<CODE BEGINS> file "ietf-http-client@2024-03-16.yang"

```
module ietf-http-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-http-client";
  prefix httpc;

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

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

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-udp-client {
    prefix udpc;
    reference
      "RFC JJJJ: YANG Groupings for UDP Clients and UDP Servers";
  }

  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module defines reusable groupings for HTTP clients that
     can be used as a basis for specific HTTP client instances.

     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 GGGG
     (https://www.rfc-editor.org/info/rfcGGGG); 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 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
  }

  // Features

  feature basic-auth {
    description
      "Indicates that the server supports configuring HTTP
       clients to authenticate themselves to an HTTP server
       using the 'basic' HTTP authentication scheme.";
    reference
      "RFC 7617: The 'Basic' HTTP Authentication Scheme";
  }
```

```
    feature tcp-supported {
      description
        "Indicates that the server supports configuring
         HTTP 1.1/2.0 clients to initiate HTTP 1.1/2.0
         connections over TCP.";
      reference
        "RFC 9110: HTTP Semantics";
    }

    feature tls-supported {
      description
        "Indicates that the server supports configuring
         HTTP 1.1/2.0 clients to initiate HTTP 1.1/2.0
         connections over TLS.";
      reference
        "RFC 9110: HTTP Semantics";
    }

    feature quic-supported {
      description
        "Indicates that the server supports configuring
         HTTP/3 clients to initiate connections over QUIC.";
      reference
        "RFC 9114: HTTP/3";
    }

    feature proxy-connect {
      description
        "Indicates that the server supports configuring HTTP
         clients to connect to a remote HTTP server via a
         proxy.";
    }

    feature http-connect-proxy {
      if-feature proxy-connect;
      description
        "Indicates that the server supports configuring
         HTTP clients to proxy clients through an HTTP
         connect proxy, , per Section 9.3.6 of RFC 9110.";
      reference
        "RFC 9110: HTTP Semantics";
    }

    feature https-connect-proxy {
      if-feature proxy-connect;
      description
        "Indicates that the server supports configuring
         HTTP clients to proxy clients through an HTTPS
```

```
          connect proxy, , per Section 9.3.6 of RFC 9110.";
      reference
        "RFC 9110: HTTP Semantics";
    }

    // Groupings

    grouping http-client-identity-grouping {
      description
        "A grouping to provide HTTP credentials used by the
         client to authenticate itself to the HTTP server.";
      container client-identity {
        nacm:default-deny-write;
        presence
          "Indicates that a client identity has been configured.
           This statement is present so the mandatory descendant
           nodes do not imply that this node must be configured.";
        description
          "The identity the HTTP client should use when
           authenticating itself to the HTTP server.";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst available authentication types.";
          case basic {
            container basic {
              if-feature "basic-auth";
              leaf user-id {
                type string;
                mandatory true;
                description
                  "The user-id for the authenticating client.";
              }
              uses ct:password-grouping {
                description
                  "The password for the authenticating client.";
              }
              description
                "The 'basic' HTTP scheme credentials.";
              reference
                "RFC 7617: The 'Basic' HTTP Authentication Scheme";
            }
          }
        }
      }
    } // grouping http-client-identity-grouping

    grouping http-client-grouping {
```

```
      description
        "A reusable grouping for configuring a HTTP client.

         This grouping is expected to be used in conjunction with
         other configurations providing, e.g., the hostname or IP
         address and port number the client initiates connections
         to.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'http-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      uses http-client-identity-grouping;

      container proxy-connect {
        nacm:default-deny-write;
        if-feature "proxy-connect";
        presence
          "Indicates that a proxy server connections have been
           configured.  This statement is present so the mandatory
           descendant nodes do not imply that this node must be
           configured.";
        description
          "Configures the proxy server the HTTP-client is to
           connect through.";
        choice proxy-type {
          mandatory true;
          description
            "Choice amongst proxy server types.";
          case http {
            if-feature http-connect-proxy;
            container http-connect {
              description
                "Container for HTTP Proxy (Web Proxy) server
                 configuration parameters, per Section 9.3.6
                 of RFC 9110.";
              reference
                "RFC 9110: HTTP Semantics";
              container tcp-client-parameters {
                description
                  "TCP client parameters.";
                uses tcpc:tcp-client-grouping {
                  refine "remote-port" {
```

```
                    default "80";
                    description
                      "The HTTP client will attempt to connect
                       to the IANA-assigned well-known port for
                       'http' (80) if no value is specified.";
                  }
                }
              }
              container http-client-parameters {
                description
                  "HTTP client parameters.";
                uses http-client-identity-grouping;
              }
            }
          }
          case https {
            if-feature https-connect-proxy;
            container https-connect {
              description
                "Container for HTTPS Proxy (Secure Web Proxy)
                 server configuration parameters, per Section
                 9.3.6 of RFC 9110.";
              reference
                "RFC 9110: HTTP Semantics";
              container tcp-client-parameters {
                description
                  "TCP client parameters.";
                uses tcpc:tcp-client-grouping {
                  refine "remote-port" {
                    default "443";
                    description
                      "The HTTP client will attempt to connect
                       to the IANA-assigned well-known port for
                       'https' (443) if no value is specified.";
                  }
                }
              }
              container tls-client-parameters {
                description
                  "TLS client parameters.";
                uses tlsc:tls-client-grouping;
              }
              container http-client-parameters {
                description
                  "HTTP client parameters.";
                uses http-client-identity-grouping;
              }
            }
```

```
          }
        }
      }
    } // grouping http-client-grouping

    grouping http-client-stack-grouping {
      description
        "A grouping that defines common HTTP-based protocol stacks.";
      choice transport {
        mandatory true;
        description
          "Choice amongst various transports type.  TCP, with and
           without TLS are defined here, with 'feature' statements
           so that they may be disabled.  Other transports MAY be
           augmented in as 'case' statements by future efforts.";
        case tcp {
          if-feature "tcp-supported";
          container tcp {
            description
              "Container for TCP-based HTTP protocols.";
            container tcp-client-parameters {
              description
                "TCP client parameters.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "80";
                  description
                    "The HTTP client will attempt to connect
                     to the IANA-assigned well-known port for
                     'http' (80) if no value is specified.";
                }
              }
            }
            container http-client-parameters {
              description
                "HTTP client parameters.";
              uses http-client-grouping;
            }
          }
        }
        case tls {
          if-feature "tls-supported";
          container tls {
            description
              "Container for TLS-based HTTP protocols.";
            container tcp-client-parameters {
              description
                "TCP client parameters.";
```

```
            uses tcpc:tcp-client-grouping {
              refine "remote-port" {
                default "443";
                description
                  "The HTTP client will attempt to connect
                   to the IANA-assigned well-known port for
                   'https' (443) if no value is specified.";
              }
            }
          }
          container tls-client-parameters {
            description
              "TLS client parameters.";
            uses tlsc:tls-client-grouping;
          }
          container http-client-parameters {
            description
              "HTTP client parameters.";
            uses http-client-grouping;
          }
        }
      }
      case quic {
        if-feature "quic-supported";
        container quic {
          description
            "Container for the QUIC-based HTTP/3 protocols.";
          container udp-client-parameters {
            description
              "UDP client parameters.";
            uses udpc:udp-client-grouping;
          }
          container tls-client-parameters {
            description
              "TLS client parameters.";
            uses tlsc:tls-client-grouping;
          }
          container http-client-parameters {
            description
              "HTTP client parameters.";
            uses http-client-grouping;
          }
        }
      }
    }
  } // http-client-stack-grouping

}
```

<CODE ENDS>

3. The "ietf-http-server" Module

This section defines a YANG 1.1 module called "ietf-http-server". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-http-server" module in terms of its features and groupings.

3.1.1. Features

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

```
Features:
  +-- client-auth-supported
  +-- local-users-supported
  +-- basic-auth
  +-- tcp-supported
  +-- tls-supported
```

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

3.1.2. Groupings

The "ietf-http-server" module defines the following "grouping" statements:

* http-server-grouping
* http-server-stack-grouping

Each of these groupings are presented in the following subsections.

3.1.2.1. The "http-server-grouping" Grouping

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

```
  grouping http-server-grouping:
    +-- server-name?             string
    +-- client-authentication! {client-auth-supported}?
       +-- users {local-users-supported}?
          +-- user* [user-id]
             +-- user-id?       string
             +-- (auth-type)
                +--:(basic)
                   +-- basic {basic-auth}?
                      +-- username?   string
                      +-- password
                         +-- hashed-password?   ianach:crypt-hash
                         +--ro last-modified?     yang:date-and-time
```

Comments:

* The "http-server-grouping" defines the configuration for just the "HTTP" part of a protocol stack. It does not, for instance, define any configuration for the "TCP" or "TLS" protocol layers (for that, see Section 3.1.2.2).

* The "server-name" node defines the HTTP server's name, as presented to HTTP clients.

* The "client-authentication" node, which must by enabled by a feature, defines a very simple user-database. Only the "basic" authentication scheme is supported, albeit it must be enabled by a "feature". Other authentication schemes MAY be augmented in.

3.1.2.2. The "http-server-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "http-server-stack-grouping" grouping:

```
  grouping http-server-stack-grouping:
    +-- (transport)
       +--:(tcp) {tcp-supported}?
       |  +-- tcp
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-server-parameters
       |        +---u http-server-grouping
       +--:(tls) {tls-supported}?
       |  +-- tls
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- tls-server-parameters
       |     |  +---u tlss:tls-server-grouping
       |     +-- http-server-parameters
       |        +---u http-server-grouping
       +--:(quic) {quic-supported}?
          +-- quic
             +-- udp-server-parameters
             |  +---u udps:udp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
                +---u http-server-grouping
```

Comments:

* The "http-server-stack-grouping" is a convenience grouping for consuming modules. It defines protocol stacks for HTTP/1.1, HTTP/2, and HTTP/3, with each option enabled by a "feature" statement for application control. Other protocols may be added by future work using the YANG "augment" statement.

* For the referenced grouping statement(s):

  - The "udp-server-grouping" grouping is discussed in Section 3 of [I-D.ietf-netconf-udp-client-server].
  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-http-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not define any protocol-accessible nodes.

3.2. Example Usage

This section presents an example showing the http-server-grouping populated with some data.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<http-server xmlns="urn:ietf:params:xml:ns:yang:ietf-http-server">
  <server-name>foo.example.com</server-name>
</http-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC7317], [RFC7617], [RFC9110], [I-D.ietf-netconf-tcp-client-server], and [I-D.ietf-netconf-tls-client-server].

```
<CODE BEGINS> file "ietf-http-server@2024-03-16.yang"

module ietf-http-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-http-server";
  prefix https;

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

  import iana-crypt-hash {
    prefix ianach;
    reference
      "RFC 7317: A YANG Data Model for System Management";
  }

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

```
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-server {
    prefix tlss;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-udp-server {
    prefix udps;
    reference
      "RFC JJJJ: YANG Groupings for UDP Clients and UDP Servers";
  }

  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module defines reusable groupings for HTTP servers that
     can be used as a basis for specific HTTP server instances.

     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 GGGG
     (https://www.rfc-editor.org/info/rfcGGGG); 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 2024-03-16 {
        description
          "Initial version";
        reference
          "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
      }

      // Features

      feature client-auth-supported {
        description
          "Indicates that the server supports configuring HTTP
           servers to authenticate HTTP clients.  HTTP-level client
           authentication may not be needed when client authentication
           is expected to occur only at another protocol layer (e.g.,
           TLS).";
      }

      feature local-users-supported {
        if-feature "client-auth-supported";
        description
          "Indicates that the server supports configuring client
           authentication with its own database of local users, as
           opposed to in an application specific location.";
      }

      feature basic-auth {
        if-feature "local-users-supported";
        description
          "Indicates that the server supports configuring 'basic'
           authentication credentials in its local user database.";
        reference
          "RFC 7617: The 'Basic' HTTP Authentication Scheme";
      }

      feature tcp-supported {
        description
          "Indicates that the server supports configuring HTTP
           servers to listen for HTTP 1.1/2.0 connections over TCP.";
        reference
          "RFC 9110: HTTP Semantics";
      }
```

```
  feature tls-supported {
    description
      "Indicates that the server supports configuring HTTP
       servers to listen for HTTP 1.1/2.0 connections over TLS.";
    reference
      "RFC 9110: HTTP Semantics";
  }

  feature quic-supported {
    description
      "Indicates that the server supports configuring HTTP
       servers to listen for HTTP/3 connections over QUIC.";
    reference
      "RFC 9114: HTTP/3";
  }

  // Groupings

  grouping http-server-grouping {
    description
      "A reusable grouping for configuring an HTTP server.

       Note that this grouping uses fairly typical descendant
       node names such that a stack of 'uses' statements will
       have name conflicts.  It is intended that the consuming
       data model will resolve the issue (e.g., by wrapping
       the 'uses' statement in a container called
       'http-server-parameters').  This model purposely does
       not do this itself so as to provide maximum flexibility
       to consuming models.";

    leaf server-name {
      nacm:default-deny-write;
      type string;
      description
        "The value of the 'Server' header field.  If not set, then
         underlying software's default value is used.  Set to the
         empty string to disable.";
    }

    container client-authentication {
      if-feature "client-auth-supported";
      nacm:default-deny-write;
      presence
        "Indicates that HTTP based client authentication is
         configured.  This statement is present so the mandatory
         descendant nodes do not imply that this node must be
         configured.";
```

```
        description
          "Configures how the HTTP server can authenticate HTTP
           clients.  The HTTP server will request that the HTTP
           client send authentication when needed.";
        container users {
          if-feature "local-users-supported";
          description
            "A list of locally configured users.";
          list user {
            key "user-id";
            description
              "The list of local users configured on this device.";
            leaf user-id {
              type string;
              description
                "The user-id for the authenticating client.";
            }
            choice auth-type {
              mandatory true;
              description
                "The authentication type.";
              case basic {
                container basic {
                  if-feature "basic-auth";
                  leaf username {
                    type string;
                    description
                      "The username for the authenticating HTTP
                       client.";
                  }
                  container password {
                    description
                      "The hashed password the HTTP server uses to
                       authenticate this user.  A user is authenticated
                       if the hash of the supplied password matches
                       this value.";
                    leaf hashed-password {
                      type ianach:crypt-hash;
                      description
                        "The password for the authenticating client.";
                    }
                    leaf last-modified {
                      type yang:date-and-time;
                      config false;
                      description
                        "Identifies when the password was last set.";
                    }
                  }
```

```
                  description
                    "The 'basic' HTTP scheme credentials.";
                  reference
                    "RFC 7617:
                      The 'Basic' HTTP Authentication Scheme";
                }
              }
            }
          }
        }
      } // container client-authentication
    } // grouping http-server-grouping

    grouping http-server-stack-grouping {
      description
        "A grouping that defines common HTTP-based protocol stacks.";
      choice transport {
        mandatory true;
        description
          "Choice amongst various transports type.  TCP, with and
           without TLS are defined here, with 'feature' statements
           so that they may be disabled.  Other transports MAY be
           augmented in as 'case' statements by future efforts.";
        case tcp {
          if-feature "tcp-supported";
          container tcp {
            description
              "Container for TCP-based HTTP protocols.";
            container tcp-server-parameters {
              description
                "TCP-level server parameters to
                 listen for HTTP connections.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "80";
                  description
                    "The HTTP client will attempt to connect
                     to the IANA-assigned well-known port for
                     'http' (80) if no value is specified.";
                }
              }
            }
            container http-server-parameters {
              description
                "HTTP-level server parameters to
                 listen for HTTP connections.";
              uses http-server-grouping;
            }
```

```
          }
        }
        case tls {
          if-feature "tls-supported";
          container tls {
            description
              "Container for TLS-based HTTP protocols.";
            container tcp-server-parameters {
              description
                "TCP-level server parameters to
                 listen for HTTPS connections.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "443";
                  description
                    "The HTTP client will attempt to connect
                     to the IANA-assigned well-known port for
                     'https' (443) if no value is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "TLS-level server parameters to
                 listen for HTTPS connections.";
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "HTTP-level server parameters to
                 listen for HTTPS connections.";
              uses http-server-grouping;
            }
          }
        }
        case quic {
          if-feature "quic-supported";
          container quic {
            description
              "Container for the QUIC-based HTTP/3 protocol.";
            container udp-server-parameters {
              description
                "UDP-level server parameters.";
              uses udps:udp-server-grouping;
            }
            container tls-server-parameters {
              description
                "TLS-level server parameters.";
```

```
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "HTTP-level server parameters.";
              uses http-server-grouping;
            }
          }
        }
      }
    } // http-server-stack-grouping

  }
```

<CODE ENDS>

4. Security Considerations

The two YANG modules in this document define groupings and will not be deployed as standalone modules. Their security implications may be context dependent based on their use in other modules. The designers of modules which import these grouping must conduct their own analysis of the security considerations.

4.1. Considerations for the "ietf-http-client" YANG Module

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

The "ietf-http-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

The follwing writable data nodes defined in this YANG module, i.e., excluding imported modules, are considered sensitive or vulnerable in network environments:

* The "client-identity" node in the "http-client-identity-grouping" grouping may be considered sensitive or vulnerable in some network environments. For this reason, the NACM extension "default-deny-write" has been applied to it. A misconfigured "client-identity" node may result in loss of connectivity or, perhaps, unexpectedly elevated authorization.

* The "proxy-connect" node in the "http-client-grouping" grouping may be considered sensitive or vulnerable in some network environments. For this reason, the NACM extension "default-deny-write" has been applied to it. A misconfigured "proxy-connect" node may result in loss of connectivity or, perhaps, unexpectedly elevated authorization.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

4.2. Considerations for the "ietf-http-server" YANG Module

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

The "ietf-http-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

The follwing writable data nodes defined in this YANG module, i.e., excluding imported modules, are considered sensitive or vulnerable in network environments:

* The "server-name" node in the "http-server-grouping" grouping may be considered sensitive or vulnerable in some network environments. For this reason, the NACM extension "default-deny-write" has been applied to it. A misconfigured "server-name" may mislead clients into not knowing how to interoperate with the server (e.g., "foo v1.0" vs "foo 2.0").

* The "client-authentication" node in the "http-server-grouping" grouping may be considered sensitive or vulnerable in some network environments. For this reason, the NACM extension "default-deny-write" has been applied to it. The feature "tls-supported" is not required, in order to support cases where an external device is used to terminate TLS connections, but such arrangements leave the client- credentials to be unprotected by the transport. Misconfigured "client-authentication" may lead the server to authenticate invalid client credentials.

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 two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-http-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-http-client
prefix:       httpc
reference:    RFC GGGG

name:         ietf-http-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-http-server
prefix:       https
reference:    RFC GGGG
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[I-D.ietf-netconf-udp-client-server]
Feng, A. H., Francois, P., and K. Watsen, "YANG Groupings for UDP Clients and UDP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-udp-client-server-01, 27 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-udp-client-server-01>.

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/info/rfc7617>.

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

6.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-

netconf-http-client-server-19, 1 March 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-05, 21 February 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

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

[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>.

[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>.

Appendix A. Change Log

A.1. 00 to 01

* Modified Abstract and Intro to be more accurate wrt intended applicability.

* In ietf-http-client, removed "protocol-version" and all auth schemes except "basic".

* In ietf-http-client, factored out "client-identity-grouping" for proxy connections.

* In ietf-http-server, removed "choice required-or-optional" and "choice inline-or-external".

* In ietf-http-server, moved the basic auth under a "choice auth-type" limited by new "feature basic-auth".

A.2. 01 to 02

* Removed the unused "external-client-auth-supported" feature from ietf-http-server.

A.3. 02 to 03

* Removed "protocol-versions" from ietf-http-server based on HTTP WG feedback.

* Slightly restructured the "proxy-server" definition in ietf-http-client.

* Added http-client example show proxy server use.

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

A.4. 03 to 04

* Added a parent "container" to "client-identity-grouping" so that it could be better used by the proxy model.

* Added a "choice" to the proxy model enabling selection of proxy types.

* Added 'http-client-stack-grouping' and 'http-server-stack-grouping' convenience groupings.

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

* Updated the Security Considerations section.

A.5. 04 to 05

* Fixed titles and a ref in the IANA Considerations section

* Cleaned up examples (e.g., removed FIXMEs)

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

* Updated the "ietf-http-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

A.6. 05 to 06

* Removed note questioning if okay for app to augment-in a 'path' node when needed, discussed during the 108 session.

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.7. 06 to 07

* Added XML-comment above examples explaining the reason for the unusual top-most element's presence.
* Renamed 'client-auth-config-supported' to 'client-auth-supported' consistent with other drafts.
* Wrapped 'container basic' choice inside a 'case basic' per best practice.
* Aligned modules with `pyang -f` formatting.
* Fixed nits found by YANG Doctor reviews.

A.8. 07 to 08

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
* Minor editorial nits

A.9. 08 to 09

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)
* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

A.10. 09 to 10

* NO UPDATE.

A.11. 10 to 11

* Updated per Shepherd reviews impacting the suite of drafts.

A.12. 11 to 12

* Updated per Shepherd reviews impacting the suite of drafts.

A.13. 12 to 13

* Updated per Tom Petch reviews.
* Renamed draft title to limit to HTTP 1.1 and 2.0.
* Added refs to RFCs 7317, 7617, and 9110.
* Added "if-feature local-users-supported" to "feature basic-auth".

A.14. 13 to 14

* Addresses AD review comments.
* Added note to Editor to fix line foldings.
* Removed "Conventions" section as there are no "BASE64VALUE=" values used in draft.
* Clarified that the modules, when implemented, do not define any protocol-accessible nodes.
* Added Security Considerations text to also look a SC-section from imported modules.
* Removed "A wrapper around the foobar parameters to avoid name collisions" text.
* Removed "public-key-format" and "public-key" nodes from examples.

A.15. 14 to 15

* Addresses AD review by Rob Wilton.

A.16. 15 to 16

* Addresses 1st-round of IESG reviews.

A.17. 16 to 18

* Addresses issues found in OpsDir review of the ssh-client-server draft.
* s/defines/presents/ in a few places.
* Add refs to where the 'operational' and 'system' datastores are defined.
* Renamed Security Considerations section s/Template for/ Considerations for/.
* Updated "http-client-stack-grouping" and "http-server-stack-grouping" for HTTP/3.

A.18. 18 to 19

* Address IESG review comments.

A.19. 19 to 20

* Updated to reflect comments from Paul Wouters.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Ben Schwartz, Éric Vyncke, Mark Nottingham, Mahesh Jethanandani, Murray Kucherawy, Orie Steele, Rob Wilton, Roman Danyliw, Shivan Sahib, and Willy Tarreau.

Author's Address

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

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2021

M. Jethanandani
Kloud Services
K. Watsen
Watsen Networks

July 9, 2020

An HTTPS-based Transport for Configured Subscriptions
draft-ietf-netconf-https-notif-03

Abstract

This document defines a YANG data module for configuring HTTPS based configured subscription, as defined in RFC 8639. The use of HTTPS maximizes transport-level interoperability, while allowing for encoding selection from text, e.g. XML or JSON, to binary.

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 January 10, 2021.

Table of Contents

1. Introduction

Subscription to YANG Notifications [RFC8639] defines a YANG data module for configuring subscribed notifications. It defines a "subscriptions" container that contains a list of receivers, but it defers the configuration and management of those receivers to other documents. This document defines two YANG 1.1 [RFC7950] data modules, one for augmenting the Subscription to YANG Notifications [RFC8639] to add a transport type, and another for configuring and managing HTTPS based receivers for the notifications.

The first module allows for different transports to be configured for the same receiver instance. The second module describes how to enable the transmission of YANG modeled notifications, in the configured encoding (i.e., XML, JSON) over HTTPS. Notifications are delivered in the form of a HTTPS POST. The use of HTTPS maximizes transport-level interoperability, while the encoding selection pivots between implementation simplicity (XML, JSON) and throughput (text versus binary).

Configured subscriptions enable a server, acting as a publisher of notifications, to proactively push notifications to external receivers without the receivers needing to first connect to the server, as is the case with dynamic subscriptions.

1.1. Applicability Statement

While the YANG modules have been defined as an augmentation of Subscription to YANG Notifications [RFC8639], the notification method defined in this document MAY be used outside of Subscription to YANG Notifications [RFC8639] by using some of the definitions from this module along with the grouping defined in Groupings for HTTP Clients and Servers [I-D.ietf-netconf-http-client-server]. For an example on how that can be done, see Section 8.2.

1.2. Note to RFC Editor

This document uses several placeholder values throughout the document. Please replace them as follows and remove this section before publication.

RFC XXXX, where XXXX is the number assigned to this document at the time of publication.

2020-07-10 with the actual date of the publication of this document.

1.3. Abbreviations

Acronym	Expansion
HTTP	Hyper Text Transport Protocol
HTTPS	Hyper Text Transport Protocol Secure
TCP	Transmission Control Protocol
TLS	Transport Layer Security

1.4. Terminology

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

1.4.1. Subscribed Notifications

The following terms are defined in Subscription to YANG Notifications [RFC8639].

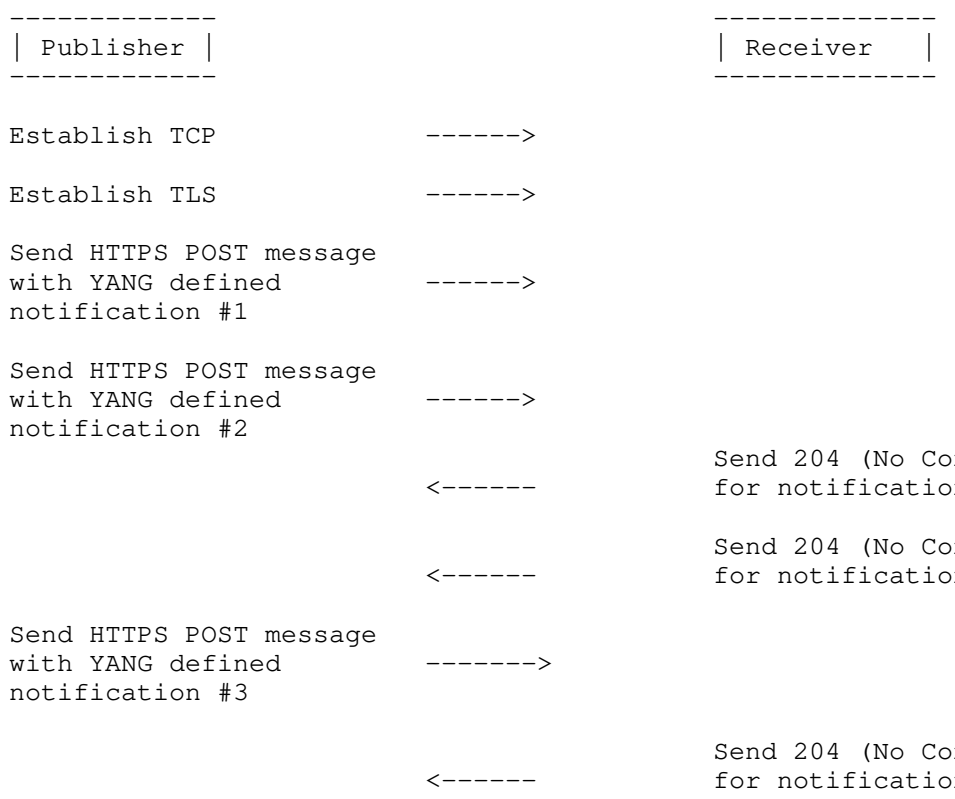
- Subscribed Notifications

1.5. Receiver and Publisher Interaction

The interaction between the receiver and the publisher can be of type "pipelining" or send multiple notifications as part of a "bundled-message", as defined in Notification Message Headers and Bundles [I-D.ietf-netconf-notification-messages]

1.5.1. Pipelining of messages

In the case of "pipelining", the flow of messages would look something like this.



The content of the exchange would look something like this.

Request:

```
POST /some/path HTTP/1.1
Host: my-receiver.my-domain.com
Content-Type: application/yang-data+xml

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-03-22T12:35:00Z</eventTime>
  <foo xmlns="https://example.com/my-foobar-module">
    ...
  </foo>
</notification>

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-03-22T12:35:00Z</eventTime>
  <bar xmlns="https://example.com/my-foobar-module">
    ...
  </bar>
</notification>

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-03-22T12:35:01Z</eventTime>
  <baz xmlns="https://example.com/my-foobar-module">
    ...
  </baz>
</notification>
```

Response:

```
HTTP/1.1 204 No Content
Date: Fri, 03 Mar 2019 12:35:00 GMT
Server: my-receiver.my-domain.com


HTTP/1.1 204 No Content
Date: Fri, 03 Mar 2019 12:35:00 GMT
Server: my-receiver.my-domain.com

HTTP/1.1 204 No Content
Date: Fri, 03 Mar 2019 12:35:01 GMT
Server: my-receiver.my-domain.com
```

2. Learning Receiver Capabilities

2.1. Introduction

To learn the capabilities of the receiver, the publisher can issue a HTTPS GET request with Accept-Type set to application/ietf-https-notif-cap+xml or application/ietf-https-notif-cap+json, with latter as the mandatory to implement, and the default in case the type is not specified. If the receiver supports capabilities such as binary encoding of data, it can return that as a capability in a response. Please note that, when used in conjunction with Subscription to YANG Notifications [RFC8639], dynamic discovery of the receiver's supported encoding is considered only when the "/subscriptions/subscription/encoding" leaf is not configured, per the "encoding" leaf's description statement.

2.2. Example

The publisher can send the following request to learn the receiver capabilities. The Accept-Type states its preferred order for Content-Type that it wants to receive starting with XML, and if not supported, to use JSON encoding. Currently, there is only one capability of binary encoding defined.

```
GET / HTTP/1.1
Host: example.com
Accept-Type: application/ietf-https-notif-cap+xml, application/ietf-https-notif-c
ap+json
```

In case the receiver supports the first Accept-Type, its response should look like this:

```
HTTP/1.1 200 OK
Date: Wed, 26 Feb 2020 20:33:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/ietf-https-notif-cap+xml
Content-Length: nnn

<receiver-capabilities>
  <receiver-capability>
    <urn:ietf:params:https-config:capability:binary-encoding:1.0>
  </receiver-capability>
</receiver-capabilities>
```

3. The "ietf-sub-notif-recv-list" Module

3.1. Data Model Overview

This YANG module augments ietf-subscribed-notifications module to define a choice of transport types that other modules such as the ietf-https-notif module can use to define a transport specific receiver.

```
module: ietf-sub-notif-recv-list
  augment /sn:subscriptions:
    +--rw receiver-instances
       +--rw receiver-instance* [name]
          +--rw name    string
          +--rw (transport-type)
  augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
    +--rw receiver-instance-ref?   leafref
```

3.2. YANG Module

```
<CODE BEGINS> file "ietf-sub-notif-recv-list@2020-07-10.yang"
module ietf-sub-notif-recv-list {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sub-notif-recv-list";
  prefix "snrl";

  import ietf-subscribed-notifications {
    prefix sn;

    reference
      "I-D.ietf-netconf-subscribed-notifications";
  }

  organization
    "IETF NETCONF Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/netconf>
     WG List:  <netconf@ietf.org>

     Authors: Mahesh Jethanandani (mjethanandani at gmail dot com)
              Kent Watsen (kent plus ietf at watsen dot net)";
  description
    "YANG module for augmenting Subscribed Notifications to add
     a transport type.
```

```
    Copyright (c) 2018 IETF Trust and the persons identified as
    the document authors.  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
    (http://trustee.ietf.org/license-info).

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

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

  revision "2020-07-10" {
    description
      "Initial Version.";
    reference
      "RFC XXXX, YANG Data Module for HTTPS Notifications.";
  }

  augment "/sn:subscriptions" {
    container receiver-instances {
      description
        "A container for all instances of receivers.";

      list receiver-instance {
        key "name";

        leaf name {
          type string;
          description
            "An arbitrary but unique name for this receiver instance.";
        }

        choice transport-type {
          mandatory true;
          description
            "Choice of different types of transports used to send
             notifications.";
        }
        description
          "A list of all receiver instances.";
      }
```

```
    }
    description
      "Augment the subscriptions container to define the transport
       type.";
  }

  augment "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
    leaf receiver-instance-ref {
      type leafref {
        path "/sn:subscriptions/snrl:receiver-instances/" +
             "snrl:receiver-instance/snrl:name";
      }
      description
        "Reference to a receiver instance.";
    }
    description
      "Augment the subscriptions container to define an optional
       reference to a receiver instance.";
  }

}
<CODE ENDS>
```

4. The "ietf-https-notif" Module

4.1. Data Model Overview

This YANG module is a definition of a set of receivers that are interested in the notifications published by the publisher. The module contains the TCP, TLS and HTTPS parameters that are needed to communicate with the receiver. The module augments the ietf-sub-notif-recv-list module to define a transport specific receiver. As mentioned earlier, it uses POST method to deliver the notification. The attribute 'path' defines the path for the resource on the receiver, as defined by 'path-absolute' in URI Generic Syntax [RFC3986]. The user-id used by Network Configuration Access Control Model [RFC8341], is that of the receiver and is derived from the certificate presented by the receiver as part of 'receiver-identity'.

An abridged tree diagram representing the module is shown below.

```
module: ietf-https-notif
  augment /sn:subscriptions/snrl:receiver-instances
            /snrl:receiver-instance/snrl:transport-type:
    +--:(https)
       +--rw https-receiver
          +--rw (transport)
          |  +--:(tcp) {tcp-supported,not httpc:tcp-supported}?
          |  |     ...
          |  +--:(tls) {tls-supported}?
          |        ...
          +--rw receiver-identity
             +--rw cert-maps
                   ...
```

4.2. YANG module

The YANG module imports Common YANG Data Types [RFC6991], A YANG Data Model for SNMP Configuration [RFC7407], JSON Encoding of Data Modeled with YANG [RFC7951], and Subscription to YANG Notifications [RFC8639].

The YANG module is shown below.

```
<CODE BEGINS> file "ietf-https-notif@2020-07-10.yang"
module ietf-https-notif {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-https-notif";
  prefix "hn";

  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "I-D.ietf-netconf-subscribed-notifications";
  }

  import ietf-http-client {
    prefix httpc;

    reference
      "I-D.ietf-netconf-http-client-server";
  }

  import ietf-sub-notif-recv-list {
    prefix snrl;

    reference
      "RFC XXXX, YANG Data Module for HTTPS Notifications.";
```

```
  }

  import ietf-x509-cert-to-name {
    prefix x509c2n;

    reference
      "RFC 7407: YANG Data Model for SNMP Configuration.";
  }

  organization
    "IETF NETCONF Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/netconf>
     WG List:  <netconf@ietf.org>

     Authors: Mahesh Jethanandani (mjethanandani at gmail dot com)
              Kent Watsen (kent plus ietf at watsen dot net)";
  description
    "YANG module for configuring HTTPS base configuration.

     Copyright (c) 2018 IETF Trust and the persons identified as
     the document authors.  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
     (http://trustee.ietf.org/license-info).

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

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

  revision "2020-07-10" {
    description
      "Initial Version.";
    reference
      "RFC XXXX, YANG Data Module for HTTPS Notifications.";
  }

  identity https {
    base sn:transport;
```

```
      description
        "HTTPS transport for notifications.";
    }

    augment "/sn:subscriptions/snrl:receiver-instances/" +
            "snrl:receiver-instance/snrl:transport-type" {
      case https {
        container https-receiver {
          description
            "HTTPS receiver for notification";

          uses httpc:http-client-stack-grouping {
            refine "transport/tcp" {
              // create the logical impossibility of enabling "tcp"
              // transport
              if-feature "not httpc:tcp-supported";
            }
            augment "transport/tls/tls/http-client-parameters" {
              leaf path {
                type string;
                description
                  "Relative URI to the target resource.";
              }
              description
                "Augmentation to add a path to the target resource.";
            }
          }

          container receiver-identity {
            description
              "Specifies mechanism for identifying the receiver.
               The publisher MUST NOT include any content in a
               notification that the user is not authorized to view.";

            container cert-maps {
              uses x509c2n:cert-to-name;
              description
                "The cert-maps container is used by a TLS-based HTTP
                 server to map the HTTPS client's presented X.509
                 certificate to a 'local' username. If no matching and
                 valid cert-to-name list entry is found, the publisher
                 MUST close the connection, and MUST NOT
                 not send any notifications over it.";
              reference
                "RFC 7407: A YANG Data Model for SNMP Configuration.";
            }
          }
        }
```

```
      }
      description
        "Augment the transport-type choice to define this transport.";
    }
  }
  <CODE ENDS>
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446]. The NETCONF Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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:

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

6. Receiving Event Notifications

Encoding notifications for the HTTPS notifications is the same as the encoding notifications as defined in RESTCONF [RFC8040] Section 6.4, with the following changes. Instead of saying that for JSON-encoding purposes, the module name for "notification" element will be "ietf-restconf, it will say that for JSON-encoding purposes, the module name for "notification" element will be "ietf-https-notif".

With those changes, the SSE event notification encoded JSON example that would be sent over the HTTPS notif transport would appear as follows:

```
data: {
data:   "ietf-https-notif:notification" : {
data:     "eventTime" : "2013-12-21T00:01:00Z",
data:     "example-mod:event" : {
data:       "event-class" : "fault",
data:       "reporting-entity" : { "card" : "Ethernet0" },
data:       "severity" : "major"
data:     }
data:   }
data: }
```

7. IANA Considerations

This document registers two URI, two YANG module and two Media Types.

7.1. URI Registration

in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-http-notif
URI: urn:ietf:params:xml:ns:yang:ietf-sub-notif-recv-list

Registrant Contact: The IESG. XML: N/A, the requested URI is an XML namespace.

7.2. YANG Module Name Registration

This document registers one YANG module in the YANG Module Names registry YANG [RFC6020].

name: ietf-https-notif
namespace: urn:ietf:params:xml:ns:yang:ietf-https-notif
prefix: hn
reference: RFC XXXX

name: ietf-sub-recv-list
namespace: urn:ietf:params:xml:ns:yang:ietf-sub-notif-recv-list
prefix: snrl
reference: RFC XXXX

7.3. Media Types

7.3.1. Media Type "application/ietf-https-notif-cap+xml

Type name: application

Subtype name: ietf-https-notif-cap+xml

Required parameters: None

Optional parameters: None

Encoding considerations:
    8-bit Each conceptual YANG data node is encoded according to the XML Encoding Rules and Canonical Format for the specific YANG data node type defined in YANG 1.1 [RFC7950].

Security considerations:
    Security considerations related to the generation and consumption of RESTCONF messages are discussed in Section NN of RFC XXXX.

    Additional security considerations are specific to the semantics of particular YANG data models. Each YANG module is expected to specify security considerations for the YANG data defined in that module.

Interoperability considerations: N/A

Published specification: RFC XXXX

Applications that use this media type:
    Instance document data parsers used within a protocol or automation tool that utilize YANG-defined data structures.

Fragment identifier considerations:
    Fragment identifiers for this type are not defined. All YANG data nodes are accessible as resources using the path in the request URI.

Additional information:

    Deprecated alias names for this type: N/A
    Magic number(s): N/A
    File extension(s): None
    Macintosh file type code(s): "TEXT"

Person & email address to contact for further information:
    See Author's Address section of RFC XXXX.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Author's Address section of RFC XXXX

Change controller:
    Internet Engineering Task Force (mailto:iesg@ietf.org)

Provisional registration? (standards tree only): no

7.3.2. Media Type "application/ietf-https-notif-cap+json

Type name: application

Subtype name: ietf-https-notif-cap+json

Required parameters: None

Optional parameters: None

Encoding considerations:
    8-bit Each conceptual YANG data node is encoded according to the XML Encoding Rules and Canonical Format for the specific YANG data node type defined in JSON Encoding of Data Modeled with YANG [RFC7951].

Security considerations:
    Security considerations related to the generation and consumption of RESTCONF messages are discussed in Section NN of RFC XXXX.

    Additional security considerations are specific to the semantics of particular YANG data models. Each YANG module is expected to specify security considerations for the YANG data defined in that module.

Interoperability considerations: N/A

Published specification: RFC XXXX

Applications that use this media type:
    Instance document data parsers used within a protocol or automation tool that utilize YANG-defined data structures.

Fragment identifier considerations:
    Fragment identifiers for this type are not defined. All YANG data nodes are accessible as resources using the path in the request URI.

Additional information:

    Deprecated alias names for this type: N/A
    Magic number(s): N/A

```
    File extension(s): None
    Macintosh file type code(s): "TEXT"

Person & email address to contact for further information:
    See Author's Address section of RFC XXXX.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Author's Address section of RFC XXXX

Change controller:
    Internet Engineering Task Force (mailto:iesg@ietf.org)

Provisional registration? (standards tree only): no
```

8. Examples

This section shows some examples in how the module can be used.

8.1. Subscribed Notification based Configuration

This example shows how a HTTPS client can be configured to send notifications to a receiver at address 192.0.2.1, port 443, a 'path', with server certificates, and the corresponding trust store that is used to authenticate a connection.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscriptions
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notificatio\
ns">
    <receiver-instances
        xmlns="urn:ietf:params:xml:ns:yang:ietf-sub-notif-recv-list">
      <receiver-instance>
        <name>foo</name>
        <https-receiver
            xmlns="urn:ietf:params:xml:ns:yang:ietf-https-notif"
            xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-na\
me">
          <tls>
            <tcp-client-parameters>
              <remote-address>my-receiver.my-domain.com</remote-address>
              <remote-port>443</remote-port>
            </tcp-client-parameters>
```

```
            <tls-client-parameters>
              <server-authentication>
                <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
                <server-certs>explicitly-trusted-server-certs</server-certs>
              </server-authentication>
            </tls-client-parameters>
            <http-client-parameters>
              <client-identity>
                <basic>
                  <user-id>my-name</user-id>
                  <password>my-password</password>
                </basic>
              </client-identity>
              <path>/some/path</path>
            </http-client-parameters>
          </tls>
          <receiver-identity>
            <cert-maps>
              <cert-to-name>
                <id>1</id>
                <fingerprint>11:0A:05:11:00</fingerprint>
                <map-type>x509c2n:san-any</map-type>
              </cert-to-name>
            </cert-maps>
          </receiver-identity>
        </https-receiver>
      </receiver-instance>
    </receiver-instances>
    <subscription>
      <id>6666</id>
      <stream-subtree-filter>foo</stream-subtree-filter>
      <stream>some-stream</stream>
      <receivers>
        <receiver>
          <name>my-receiver</name>
          <receiver-instance-ref
              xmlns="urn:ietf:params:xml:ns:yang:ietf-sub-notif-recv-list">\
foo</receiver-instance-ref>
        </receiver>
      </receivers>
    </subscription>
  </subscriptions>

  <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
    <certificates>
      <name>explicitly-trusted-server-certs</name>
      <description>
        Specific server authentication certificates for explicitly
```

```
        trusted servers.  These are needed for server certificates
        that are not signed by a pinned CA.
      </description>
      <certificate>
        <name>Fred Flintstone</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
    <certificates>
      <name>explicitly-trusted-server-ca-certs</name>
      <description>
        Trust anchors (i.e. CA certs) that are used to authenticate
        server connections.  Servers are authenticated if their
        certificate has a chain of trust to one of these CA
        certificates.
      </description>
      <certificate>
        <name>ca.example.com</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </truststore>
</config>
```

8.2. Non Subscribed Notification based Configuration

In the case that it is desired to use HTTPS notif outside of Subscribed Notifications, there would have to be a module to define the configuration for where and how to send the notification, such as the following:

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

```
module example-custom-module {
  yang-version 1.1;
  namespace "http://example.com/example-custom-module";
  prefix "custom";

  import ietf-http-client {
    prefix httpc;
    reference
      "I-D.ietf-netconf-http-client-server";
  }

  organization
    "Example, Inc.";
```

```
     contact
       "Support at example.com";

     description
       "Example of module not using Subscribed Notifications module.";

     revision "2020-07-10" {
       description
         "Initial Version.";
       reference
         "RFC XXXX, YANG Data Module for HTTPS Notifications.";
     }

     container example-module {
       description
         "Example of using HTTPS notif without having to
          implement Subscribed Notifications.";

       container https-receivers {
         description
           "A container of all HTTPS notif receivers.";

         list https-receiver {
           key "name";

           leaf name {
             type string;
             description
               "A unique name for the https notif receiver.";
           }

           uses httpc:http-client-stack-grouping {
             refine "transport/tcp" {
               // create the logical impossibility of enabling "tcp"
               // transport
               if-feature "not httpc:tcp-supported";
             }
             augment "transport/tls/tls/http-client-parameters" {
               leaf path {
                 type string;
                 description
                   "Relative URI to the target resource.";
               }
               description
                 "Augmentation to add a path to the target resource.";
             }
           }
           description
```

```
              "Just include the grouping from ietf-http-client to
               realize the 'HTTPS stack'.";
          }
        }
      }
    }
```

This example shows how a HTTPS client can be configured to send notifications to a receiver at address 192.0.2.1, port 443, a 'path', with server certificates, and the corresponding trust store that is used to authenticate a connection.

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

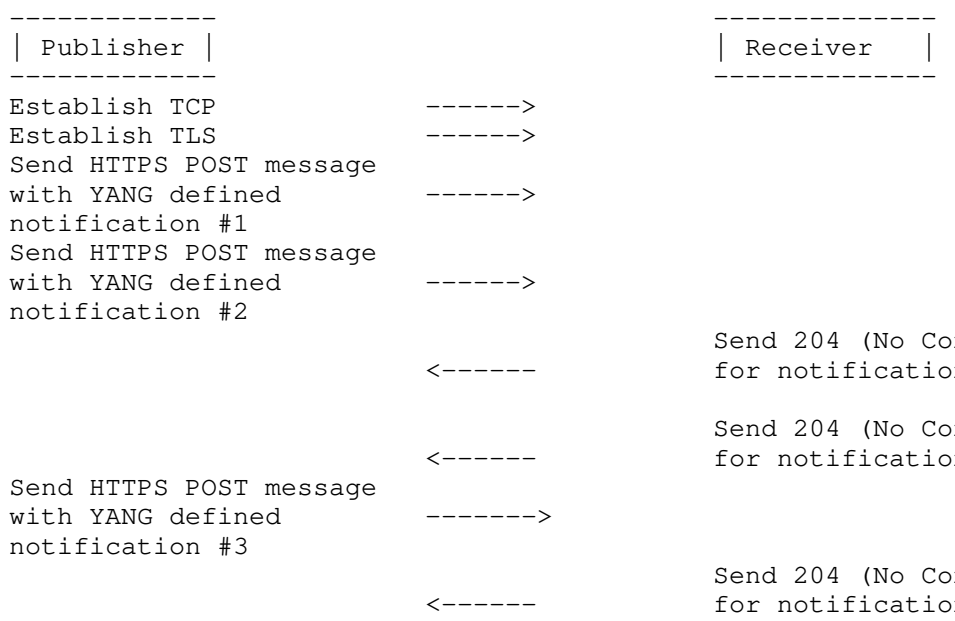
```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <example-module
      xmlns="http://example.com/example-custom-module">
    <https-receivers>
      <https-receiver>
        <name>foo</name>
        <tls>
          <tcp-client-parameters>
            <remote-address>my-receiver.my-domain.com</remote-address>
            <remote-port>443</remote-port>
          </tcp-client-parameters>
          <tls-client-parameters>
            <server-authentication>
              <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
              <server-certs>explicitly-trusted-server-certs</server-certs>
            </server-authentication>
          </tls-client-parameters>
          <http-client-parameters>
            <client-identity>
              <basic>
                <user-id>my-name</user-id>
                <password>my-password</password>
              </basic>
            </client-identity>
            <path>/some/path</path>
          </http-client-parameters>
        </tls>
      </https-receiver>
    </https-receivers>
  </example-module>

  <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
```

```
    <certificates>
      <name>explicitly-trusted-server-certs</name>
      <description>
        Specific server authentication certificates for explicitly
        trusted servers.  These are needed for server certificates
        that are not signed by a pinned CA.
      </description>
      <certificate>
        <name>Fred Flintstone</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
    <certificates>
      <name>explicitly-trusted-server-ca-certs</name>
      <description>
        Trust anchors (i.e. CA certs) that are used to authenticate
        server connections.  Servers are authenticated if their
        certificate has a chain of trust to one of these CA
        certificates.
      </description>
      <certificate>
        <name>ca.example.com</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </truststore>
</config>
```

8.3. Bundled Message

In the case of "bundled-message" as defined in Notification Message Headers and Bundles [I-D.ietf-netconf-notification-messages], something that this module supports, the flow of messages would look something like this.



The content of the exchange would look something like this.

```
   Request:
       POST /some/path HTTP/1.1
       Host: my-receiver.my-domain.com
       Content-Type: application/yang-data+xml
       <notification
         xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
         <eventTime>2019-03-22T12:35:00Z</eventTime>
         <foo xmlns="https://example.com/my-foobar-module">
           ...
         </foo>
       </notification>
       <notification
         xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
         <eventTime>2019-03-22T12:35:00Z</eventTime>
         <bar xmlns="https://example.com/my-foobar-module">
           ...
         </bar>
       </notification>
       <notification
         xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
         <eventTime>2019-03-22T12:35:01Z</eventTime>
         <baz xmlns="https://example.com/my-foobar-module">
           ...
         </baz>
       </notification>
   Response:
         HTTP/1.1 204 No Content
         Date: Fri, 03 Mar 2019 12:35:00 GMT
         Server: my-receiver.my-domain.com
         HTTP/1.1 204 No Content
         Date: Fri, 03 Mar 2019 12:35:00 GMT
         Server: my-receiver.my-domain.com
         HTTP/1.1 204 No Content
         Date: Fri, 03 Mar 2019 12:35:01 GMT
         Server: my-receiver.my-domain.com
```

9. Contributors

10. Acknowledgements

11. Normative references

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", draft-ietf-netconf-http-client-server-04 (work in progress), July 2020.

[I-D.ietf-netconf-notification-messages]
Voit, E., Jenkins, T., Birkholz, H., Bierman, A., and A. Clemm, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-08 (work in progress), November 2019.

[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>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[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>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

Authors' Addresses

Mahesh Jethanandani
Kloud Services

Email: mjethanandani@gmail.com

Kent Watsen
Watsen Networks
USA

Email: kent+ietf@watsen.net

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: 4 August 2024

M. Jethanandani
Kloud Services
K. Watsen
Watsen Networks
1 February 2024

An HTTPS-based Transport for YANG Notifications
draft-ietf-netconf-https-notif-15

Abstract

This document defines a protocol for sending asynchronous event notifications similar to notifications defined in RFC 5277, but over HTTPS. YANG modules for configuring publishers are also defined. Examples are provided illustrating how to configure various publishers.

This document requires that the publisher is a "server" (e.g., a NETCONF or RESTCONF server), but does not assume that the receiver is a server.

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 4 August 2024.

Copyright Notice

Table of Contents

1. Introduction

This document defines a protocol for sending asynchronous event notifications similar to notifications defined in NETCONF Event Notifications [RFC5277], but over HTTPS. Using HTTPS, which is a secure form of HTTP Semantics [RFC9110], maximizes transport-level interoperability, while allowing for a variety of encoding options. The protocol supports HTTP/1.1: Message Syntax and Routing [RFC9112] and, HTTP/2 [RFC9113]. While the payload does not change between these versions of HTTP and HTTP/3 [RFC9114], the underlying transport does. Since NETCONF does not support QUIC: A UDP-Based Multiplexed and Secure Transport [RFC9000], support for HTTP/3 [RFC9114], is considered out of scope of this document.

This document defines support for JSON and XML; future efforts may define support for other encodings (e.g., binary). This document requires that the publisher is a "server" (e.g., a NETCONF or RESTCONF server), but does not assume that the receiver is a NETCONF or RESTCONF server. It does expect the receiver to be an HTTPS server to receive the notifications.

This document also defines two YANG 1.1 [RFC7950] modules that extend the data model defined in Subscription to YANG Notifications [RFC8639], enabling the configuration of HTTPS-based receivers.

An example module illustrating the configuration of a publisher not using the data model defined in RFC 8639 is also provided.

Configured subscriptions enable a server (e.g., a NETCONF or RESTCONF server), acting as a publisher of notifications, to proactively push notifications to external receivers without the receivers needing to first connect to the server, as is the case with dynamic subscriptions.

1.1. Applicability Statement

While the YANG modules have been defined as an augmentation of Subscription to YANG Notifications [RFC8639], the notification method defined in this document MAY be used outside of Subscription to YANG Notifications [RFC8639] by using some of the definitions from this module along with the grouping defined in Groupings for HTTP Clients and Servers [I-D.ietf-netconf-http-client-server]. For an example on how that can be done, see Section A.2.

1.2. Note to RFC Editor

This document uses several placeholder values throughout the document. Please replace them as follows and remove this section before publication.

RFC XXXX, where XXXX is the number assigned to this document at the time of publication.

RFC YYYY, where YYYY is the number assigned to [I-D.ietf-netconf-http-client-server].

2024-02-01 with the actual date of the publication of this document.

1.3. Abbreviations

Acronym	Expansion
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
SSE	Server-Sent Events
TCP	Transmission Control Protocol
TLS	Transport Layer Security

Table 1

1.4. Terminology

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

1.4.1. Terms Imported from other RFCs

The following terms are defined in Subscription to YANG Notifications [RFC8639].

* Publisher

* Receiver

* Subscribed Notifications

The following term is defined in RESTCONF Protocol [RFC8040].

* target resource

1.5. Tree Diagram

The tree diagram for the YANG modules defined in this document use annotations defined in YANG Tree Diagrams. [RFC8340].

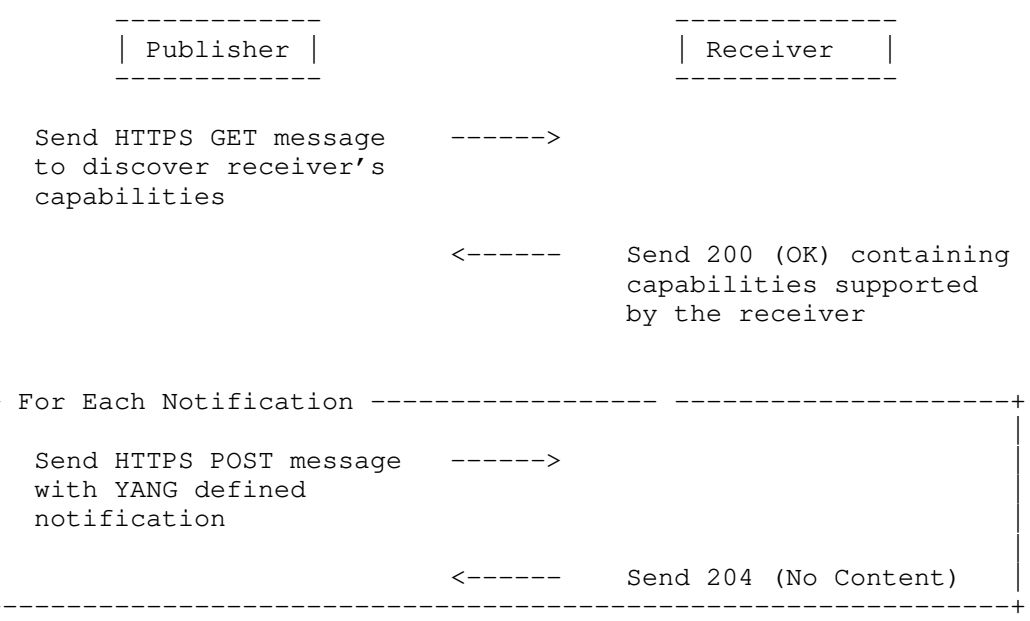
2. Overview of Publisher to Receiver Interaction

The protocol consists of two HTTP-based target resources presented by the receiver. These two resources share a common prefix that the publisher learns from a request it issues, as defined in section 3.2. If the data model in section 6.2 is used, this common prefix is defined by the "path" leaf in the "http-client-parameters" container.

* "capabilities": A target resource enabling the publisher to discover what optional capabilities a receiver supports. Publishers SHOULD query this target before sending any notifications or if ever an error occurs.

* "relay-notification": A target resource enabling the publisher to send one or more notification to a receiver. This document defines support for sending only one notification per message; a future effort MAY extend the protocol to send multiple notifications per message.

The protocol is illustrated in the diagram below:



Note that, for RFC 8639 configured subscriptions, the very first notification must be the "subscription-started" notification.

3. Discovering a Receiver's Capabilities

3.1. Applicability

For publishers using Subscription to YANG Notifications [RFC8639], dynamic discovery of a receiver's supported encoding is necessary only when the "/subscriptions/subscription/encoding" leaf is not configured, per the "encoding" leaf's description statement in the "ietf-subscribed-notification" module.

If the "encoding" leaf is not configured, and the publisher wants to send a notification in a particular format, without going through the setup operation of learning the receiver capabilities, it can do so, but has to be prepared for the case when it receives an error response, because the receiver does not support the format sent by the publisher.

3.2. Request

To learn the capabilities of a receiver, a publisher can issue an HTTPS GET request to the "capabilities" resource (see Section 2) on the receiver with "Accept" header set using the "application/xml" as defined in XML Media Types [RFC7303], and/or "application/json" as defined in JSON [RFC8259] media-types.

3.3. Response

The receiver responds with a "200 (OK)" message, having the "Content-Type" header set to either "application/xml" or "application/json" (which ever was selected), and containing in the response body a list of the receiver's capabilities encoded in the selected format.

Even though a YANG module is not defined for this interaction, the response body MUST conform to the following YANG-modeled format:

```
container receiver-capabilities {
  description
    "A container for a list of capabilities supported by
     the receiver.";
  leaf-list receiver-capability {
    type "inet:uri";
    description
      "A capability supported by the receiver.  A partial list of
       capabilities is defined in the 'Capabilities for HTTPS
       Notification Receivers' registry (see RFC XXXX). Additional
       custom capabilities MAY be defined.";
  }
}
```

As it is possible that the receiver may return custom capability URIs, the publisher MUST ignore any capabilities that it does not recognize.

3.4. Example

The publisher can send the following request to learn the receiver capabilities. In this example, the "Accept" states that the publisher wants to receive the capabilities response in XML but, if not supported, then in JSON.

```
GET /some/path/capabilities HTTP/1.1
Host: example.com
Accept: application/xml, application/json;q=0.5
```

If the receiver is able to reply using "application/xml", and assuming it is able to receive JSON and XML encoded notifications, and it is able to process the RFC 8639 state machine, the response might look like this:

```
HTTP/1.1 200 OK
Date: Wed, 26 Feb 2020 20:33:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/xml

<receiver-capabilities>
  <receiver-capability>\
    urn:ietf:capability:https-notif-receiver:encoding:json\
  </receiver-capability>
  <receiver-capability>\
    urn:ietf:capability:https-notif-receiver:encoding:xml\
  </receiver-capability>
  <receiver-capability>\
    urn:ietf:capability:https-notif-receiver:sub-notif\
  </receiver-capability>
</receiver-capabilities>
```

If the receiver is unable to reply using "application/xml", the response might look like this:

```
HTTP/1.1 200 OK
Date: Wed, 26 Feb 2020 20:33:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/json
Content-Length: nnn

{
   "receiver-capabilities": {
     "receiver-capability": [
       "urn:ietf:capability:https-notif-receiver:encoding:json",
       "urn:ietf:capability:https-notif-receiver:encoding:xml",
       "urn:ietf:capability:https-notif-receiver:sub-notif"
     ]
   }
}
```

4. Sending Event Notifications

4.1. Request

The publisher sends an HTTP POST request to the "relay-notification" resource (see Section 2) on the receiver with the "Content-Type" header set to either "application/json" or "application/xml" and a body containing the notification encoded using the specified format.

XML-encoded notifications are encoded using the format defined by NETCONF Event Notifications [RFC5277] for XML.

JSON-encoded notifications are encoded the same as specified in Section 6.4 in RESTCONF [RFC8040] with the following deviations:

* The notifications do not contain the "data:" prefix used by Server-Sent Events (SSE).

* Instead of saying that, for JSON-encoding purposes, the module name for the "notification" element is "ietf-restconf", the module name will instead be "ietf-https-notif".

4.2. Response

The response on success SHOULD be "204 (No Content)". In case of corrupted or malformed event, the response SHOULD be an appropriate HTTP error response.

4.3. Example

An XML-encoded notification might be sent as follows:

```
POST /some/path/relay-notification HTTP/1.1
Host: example.com
Content-Type: application/xml

<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-03-22T12:35:00Z</eventTime>
  <event xmlns="https://example.com/example-mod">
    <event-class>fault</event-class>
    <reporting-entity>
      <card>Ethernet0</card>
    </reporting-entity>
    <severity>major</severity>
  </event>
</notification>
```

A JSON-encoded notification might be sent as follows:

```
POST /some/path/relay-notification HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "ietf-https-notif:notification": {
    "eventTime": "2013-12-21T00:01:00Z",
    "example-mod:event" : {
      "event-class" : "fault",
      "reporting-entity" : { "card" : "Ethernet0" },
      "severity" : "major"
    }
  }
}
```

And, in either case, the response on success might be as follows:

```
HTTP/1.1 204 No Content
Date: Wed, 26 Feb 2020 20:33:30 GMT
Server: example-server
```

5. The "ietf-subscribed-notif-receivers" Module

5.1. Data Model Overview

This YANG module augments the "ietf-subscribed-notifications" module to define a choice of transport types that other modules such as the "ietf-https-notif-transport" module can use to define a transport specific receiver.

module: ietf-subscribed-notif-receivers

```
  augment /sn:subscriptions:
    +--rw receiver-instances
       +--rw receiver-instance* [name]
          +--rw name    string
          +--rw (transport-type)
  augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
    +--rw receiver-instance-ref?   leafref
```

5.2. YANG Module

The YANG module imports Subscription to YANG Notifications [RFC8639].

```
<CODE BEGINS> file "ietf-subscribed-notif-receivers@2024-02-01.yang"
module ietf-subscribed-notif-receivers {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers";
  prefix "snr";

  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: Subscription to YANG Notifications";
  }

  organization
    "IETF NETCONF Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf>
     WG List:  <netconf@ietf.org>

     Authors: Mahesh Jethanandani (mjethanandani at gmail dot com)
              Kent Watsen (kent plus ietf at watsen dot net)";

  description
    "This YANG module is implemented by Publishers implementing
     the 'ietf-subscribed-notifications' module defined in RFC 8639.

     While this module is defined in RFC XXXX, which primarily
     defines an HTTPS-based transport for notifications, this module
     is not HTTP-specific.  It is a generic extension that can be
     used by any 'notif' transport.

     This module defines two 'augment' statements.  One statement
     augments a 'container' statement called 'receiver-instances'
     into the top-level 'subscriptions' container.  The other
     statement, called 'receiver-instance-ref', augments a 'leaf'
     statement into each 'receiver' that references one of the
     afore mentioned receiver instances.  This indirection enables
     multiple configured subscriptions to send notifications to
     the same receiver instance.

     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
```

```
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; 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 "2024-02-01" {
      description
        "Initial Version.";
      reference
        "RFC XXXX: An HTTPS-based Transport for YANG Notifications.";
    }

    augment "/sn:subscriptions" {
      container receiver-instances {
        description
          "A container for all instances of receivers.";

        list receiver-instance {
          key "name";

          leaf name {
            type string;
            description
              "An arbitrary but unique name for this receiver
               instance.";
          }

          choice transport-type {
            mandatory true;
            description
              "Choice of different types of transports used to
               send notifications.  The 'case' statements must
               be augmented in by other modules.";
          }
          description
            "A list of all receiver instances.";
        }
      }
      description
        "Augment the subscriptions container to define the
         transport type.";
    }
```

```
    augment
      "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
      leaf receiver-instance-ref {
        type leafref {
          path "/sn:subscriptions/snr:receiver-instances/" +
               "snr:receiver-instance/snr:name";
        }
        description
          "Reference to a receiver instance.";
      }
      description
        "Augment the subscriptions container to define an optional
         reference to a receiver instance.";
    }
  }
  <CODE ENDS>
```

6. The "ietf-https-notif-transport" Module

6.1. Data Model Overview

This YANG module is a definition of a set of receivers that are interested in the notifications published by the publisher. The module contains the TCP, TLS and HTTPS parameters that are needed to communicate with the receiver. The module augments the "ietf-subscribed-notif-receivers" module to define a transport specific receiver.

As mentioned earlier, it uses a POST method to deliver the notification. The "http-receiver/tls/http-client-parameters/path" leaf defines the path for the resource on the receiver, as defined by "path-absolute" in URI Generic Syntax [RFC3986]. The user-id used by Network Configuration Access Control Model [RFC8341], is that of the receiver and is derived from the certificate presented by the receiver as part of "receiver-identity".

An abridged tree diagram representing the module is shown below.

```
   module: ietf-https-notif-transport

     augment /sn:subscriptions/snr:receiver-instances
               /snr:receiver-instance/snr:transport-type:
       +--:(https)
          +--rw https-receiver
             +--rw (transport)
             |  +--:(tls) {tls-supported}?
             |     +--rw tls
             |        +--rw tcp-client-parameters
             |        |  +--rw remote-address    inet:host
             |        |  +--rw remote-port?      inet:port-number
             |        |  +--rw local-address?    inet:ip-address
             |        |  |       {local-binding-supported}?
             |        |  +--rw local-port?       inet:port-number
             |        |  |       {local-binding-supported}?
             |        |  +--rw proxy-server! {proxy-connect}?
             |        |  |     ...
             |        |  +--rw keepalives! {keepalives-supported}?
             |        |        ...
             |        +--rw tls-client-parameters
             |        |  +--rw client-identity!
             |        |  |     ...
             |        |  +--rw server-authentication
             |        |  |     ...
             |        |  +--rw hello-params {tlscmn:hello-params}?
             |        |  |     ...
             |        |  +--rw keepalives {tls-client-keepalives}?
             |        |        ...
             |        +--rw http-client-parameters
             |           +--rw client-identity!
             |           |     ...
             |           +--rw proxy-connect! {proxy-connect}?
             |           |     ...
             |           +--rw path               string
             +--rw receiver-identity {receiver-identity}?
                +--rw cert-maps
                   +--rw cert-to-name* [id]
                      +--rw id             uint32
                      +--rw fingerprint    x509c2n:tls-fingerprint
                      +--rw map-type       identityref
                      +--rw name           string
```

6.2. YANG module

The YANG module imports A YANG Data Model for SNMP Configuration [RFC7407], Subscription to YANG Notifications [RFC8639], and YANG Groupings for HTTP Clients and HTTP Servers [I-D.ietf-netconf-http-client-server].

The YANG module is shown below.

```
<CODE BEGINS> file "ietf-https-notif-transport@2024-02-01.yang"
module ietf-https-notif-transport {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-https-notif-transport";
  prefix "hnt";

  import ietf-x509-cert-to-name {
    prefix x509c2n;
    reference
      "RFC 7407: YANG Data Model for SNMP Configuration.";
  }

  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: Subscription to YANG Notifications";
  }

  import ietf-subscribed-notif-receivers {
    prefix snr;
    reference
      "RFC XXXX: An HTTPS-based Transport for YANG Notifications.";
  }

  import ietf-http-client {
    prefix httpc;
    reference
      "RFC YYYY: YANG Groupings for HTTP Clients and HTTP Servers.";
  }

  organization
    "IETF NETCONF Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf>
     WG List:  <netconf@ietf.org>

     Authors: Mahesh Jethanandani (mjethanandani at gmail dot com)
              Kent Watsen (kent plus ietf at watsen dot net)";
```

```
     description
       "This YANG module is implemented by Publishers that implement
        the 'ietf-subscribed-notifications' module defined in RFC 8639.

        This module augments a 'case' statement called 'https' into
        the 'choice' statement called 'transport-type' defined
        by the 'ietf-https-notif-transport' module defined in RFC XXXX.

        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
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; 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 "2024-02-01" {
       description
         "Initial Version.";
       reference
         "RFC XXXX: An HTTPS-based Transport for YANG Notifications.";
     }

     feature receiver-identity {
       description
         "Indicates that the server supports filtering notifications
          based on the receiver's identity derived from its TLS
          certificate.";
     }

     identity https {
       base sn:transport;
       description
         "HTTPS transport for notifications.";
     }

     grouping https-receiver-grouping {
       description
```

```
        "A grouping that may be used by other modules wishing to
         configure HTTPS-based notifications without using RFC 8639.";
      uses httpc:http-client-stack-grouping {
        refine "transport/tcp" {
          // create the logical impossibility of enabling the
          // "tcp" transport (i.e., "HTTP" without the 'S').
          if-feature "not httpc:tcp-supported";
        }
        augment "transport/tls/tls/http-client-parameters" {
          leaf path {
            type string;
            mandatory true;
            description
              "A path to the target resources.  Under this
               path the receiver must support both the 'capabilities'
               and 'relay-notification' resource targets, as described
               in RFC XXXX.";
          }
          description
            "Augmentation to add a receiver-specific path for the
             'capabilities' and 'relay-notification' resources.";
        }
      }
      container receiver-identity {
        if-feature receiver-identity;
        description
          "Maps the receiver's TLS certificate to a local identity
           enabling access control to be applied to filter out
           notifications that the receiver may not be authorized
           to view.";
        container cert-maps {
          uses x509c2n:cert-to-name;
          description
            "The cert-maps container is used by a TLS-based HTTP
             server to map the HTTPS client's presented X.509
             certificate to a 'local' username. Specifically, the
             'name' field within the module is used along with
             'specified' identity to perform the match. If no
             matching and valid cert-to-name list entry is found,
             the publisher MUST close the connection, and MUST
             NOT send any notifications over it.";
          reference
            "RFC 7407: A YANG Data Model for SNMP Configuration.";
        }
      }
    }

    augment "/sn:subscriptions/snr:receiver-instances/" +
```

```
            "snr:receiver-instance/snr:transport-type" {
      case https {
        container https-receiver {
          description
            "The HTTPS receiver to send notifications to.";
          uses https-receiver-grouping;
        }
      }
      description
        "Augment the transport-type choice to include the 'https'
         transport.";
    }
  }
  <CODE ENDS>
```

7. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446]. The NETCONF Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The YANG modules in this document make use of groupings that are defined in YANG Groupings for HTTP Clients and HTTP Servers [I-D.ietf-netconf-http-client-server], YANG Groupings for TLS Clients and TLS Servers [I-D.ietf-netconf-tls-client-server], and A YANG Data Model for SNMP Configuration [RFC7407]. Please see the Security Considerations section of those documents for considerations related to sensitivity and vulnerability of the data nodes defined in them. Additionally, the parameters defined in the tls-client-grouping in the ietf-tls-client module should follow the recommendations specified in Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). [RFC9325]

There are a number of data nodes defined in the YANG modules 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 "path" node in "ietf-subscribed-notif-receivers" module can be modified by a malicious user to point to an invalid URI. Worse still, it could point the URI of their choosing, exploit the vulnerable client, and if redirects are followed to the same URI, track its usage.

  The container "receiver-identity" contains nodes like "cert-maps" that are used by the HTTP server to map to the HTTPS client's certificate to a 'local' username. An unintended modification of these nodes will result in new connection requests be denied.

Some of the readable data nodes in the YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. The model does not define any readable subtrees and data nodes that are particularly sensitive or vulnerable.

Some of the RPC operations in the YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. The model does not define any RPC operations.

8. IANA Considerations

8.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the "IETF XML" registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

8.2. The "YANG Module Names" Registry

This document registers two YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:      ietf-subscribed-notif-receivers
namespace: urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers
prefix:    snr
reference: RFC XXXX

name:      ietf-https-notif-transport
namespace: urn:ietf:params:xml:ns:yang:ietf-https-notif-transport
prefix:    hnt
reference: RFC XXXX
```

8.3. Registration of 'yang-notif' URN Sub-namespace

This document requests that IANA register a new URN Sub-namespace within the "IETF URN Sub-namespace for Registered Protocol Parameter Identifiers" registry defined in [RFC3553].

Registry Name: yang-notif
Specification: RFC XXXX
Repository: "YANG Notifications" registry

8.4. Registration of 'https' URN Sub-namespace

This document requests that IANA register a new URN Sub-namespace within the "YANG Notifications" registry group defined in [RFC3553].

Registry Name: https-capability
Specification: RFC XXXX
Repository: "Capabilities for HTTPS Notification Receivers" registry

The following note shall be at the top of the registry:

This registry defines capabilities that can be supported by HTTPS-based notification receivers.

The fields for each registry are:

* URN
  - The name of the URN (required).
  - The URN must conform to the syntax described by [RFC8141].
  - The URN must begin with the string "urn:ietf:params:yang-notif:https-capability".
* Reference
  - The RFC that defined the URN.

- The RFC must be in the form "RFC <Number>: <Title>.

* Description

  - An arbitrary description of the capability.

  - The description should be no more than a few sentences.

  - The description is to be in English, but may contain UTF-8 characters as may be needed in some cases.

The update policy is "RFC Required".

Following is the initial assignment for this registry:

```
Record:
   URN:         urn:ietf:params:yang-notif:https-capability:encoding:json
   Reference:   RFC XXXX:An HTTPS-based Transport for YANG Notifications
   Description: Identifies support for JSON-encoded notifications.

Record:
   URN:         urn:ietf:params:yang-notif:https-capability:encoding:xml
   Reference:   RFC XXXX:An HTTPS-based Transport for YANG Notifications
   Description: Identifies support for XML-encoded notifications.

Record:
   URN:         urn:ietf:params:yang-notif:https-capability:sub-notif
   Reference:   RFC XXXX:An HTTPS-based Transport for YANG Notifications
   Description: Identifies support for state machine described in
                RFC 8639, enabling the publisher to send, e.g., the
                "subscription-started" notification.
```

9. References

9.1. 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>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <https://www.rfc-editor.org/info/rfc3553>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[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>.

[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>.

[RFC7303] Thompson, H. and C. Lilley, "XML Media Types", RFC 7303, DOI 10.17487/RFC7303, July 2014, <https://www.rfc-editor.org/info/rfc7303>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[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>.

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

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

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

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/info/rfc9112>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[I-D.ietf-netconf-http-client-server] Watsen, K., "YANG Groupings for HTTP 1.1/2.0 Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-15, 26 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-15>.

[I-D.ietf-netconf-tls-client-server] Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-36, 29 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-36>.

9.2. Informative references

[RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <https://www.rfc-editor.org/info/rfc8141>.

Appendix A. Configuration Examples

This non-normative section shows two examples for how the "ietf-https-notif-transport" module can be used to configure a publisher to send notifications to a receiver.

In both examples, the publisher, being an HTTPS client, is configured to send notifications to a receiver.

A.1. Using Subscribed Notifications (RFC 8639)

This example shows how an RFC 8639 [RFC8639] based publisher can be configured to send notifications to a receiver.

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

<?xml version="1.0" encoding="UTF-8"?>
<subscriptions
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications\
">
  <receiver-instances
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-recei\
```

```
   vers">
       <receiver-instance>
         <name>global-receiver-def</name>
         <https-receiver
             xmlns="urn:ietf:params:xml:ns:yang:ietf-https-notif-transp\
   ort"
             xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-\
   to-name">
           <tls>
             <tcp-client-parameters>
               <remote-address>receiver.example.com</remote-address>
               <remote-port>443</remote-port>
             </tcp-client-parameters>
             <tls-client-parameters>
               <server-authentication>
                 <ca-certs>
                   <local-definition>
                     <certificate>
                       <name>Server Cert Issuer #1</name>
                       <cert-data>base64encodedvalue==</cert-data>
                     </certificate>
                   </local-definition>
                 </ca-certs>
               </server-authentication>
             </tls-client-parameters>
             <http-client-parameters>
               <client-identity>
                 <basic>
                   <user-id>my-name</user-id>
                   <cleartext-password>my-password</cleartext-password>
                 </basic>
               </client-identity>
               <path
                   xmlns="urn:ietf:params:xml:ns:yang:ietf-https-notif-\
   transport">/some/path</path>
             </http-client-parameters>
           </tls>
           <receiver-identity>
             <cert-maps>
               <cert-to-name>
                 <id>1</id>
                 <fingerprint>11:0A:05:11:00</fingerprint>
                 <map-type>x509c2n:san-any</map-type>
               </cert-to-name>
             </cert-maps>
           </receiver-identity>
         </https-receiver>
       </receiver-instance>
```

```
    </receiver-instances>
    <subscription>
      <id>6666</id>
      <transport xmlns:ph="urn:ietf:params:xml:ns:yang:ietf-https-noti\
  f-transport">ph:https</transport>
      <stream-subtree-filter>
        <some-subtree-filter/>
      </stream-subtree-filter>
      <stream>some-stream</stream>
      <receivers>
        <receiver>
          <name>subscription-specific-receiver-def</name>
          <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ie\
  tf-subscribed-notif-receivers">global-receiver-def</receiver-instanc\
  e-ref>
        </receiver>
      </receivers>
    </subscription>
  </subscriptions>
  <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
    <certificate-bags>
      <certificate-bag>
        <name>explicitly-trusted-server-ca-certs</name>
        <description>
          Trust anchors (i.e. CA certs) that are used to
          authenticate connections to receivers.  Receivers
          are authenticated if their certificate has a chain
          of trust to one of these CA certificates.
          certificates.
        </description>
        <certificate>
          <name>ca.example.com</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
        <certificate>
          <name>Fred Flintstone</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificate-bag>
    </certificate-bags>
  </truststore>
```

A.2. Not Using Subscribed Notifications

In the case that it is desired to use HTTPS-based notifications outside of Subscribed Notifications, an application-specific module would need to define the configuration for sending the notification.

Following is an example module. Note that the module "uses" the "https-receiver-grouping" grouping from the "ietf-https-notif-transport" module.

```
module example-custom-module {
  yang-version 1.1;
  namespace "http://example.com/example-custom-module";
  prefix "custom";

  import ietf-https-notif-transport {
    prefix "hnt";
    reference
      "RFC XXXX:
        An HTTPS-based Transport for Configured Subscriptions";
  }

  organization
    "Example, Inc.";

  contact
    "Support at example.com";

  description
    "Example of module not using Subscribed Notifications module.";

  revision "2024-02-01" {
    description
      "Initial Version.";
    reference
      "RFC XXXX: An HTTPS-based Transport for YANG Notifications.";
  }

  container example-module {
    description
      "Example of using HTTPS notif without having to
       implement Subscribed Notifications.";

    container https-receivers {
      description
        "A container of all HTTPS notif receivers.";
      list https-receiver {
       key "name";
        description
          "A list of HTTPS notif receivers.";
       leaf name {
         type string;
         description
           "A unique name for the https notif receiver.";
```

```
          }
           uses hnt:https-receiver-grouping;
         }
       }
     }
   }
```

Following is what the corresponding configuration looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<example-module xmlns="http://example.com/example-custom-module">
  <https-receivers>
    <https-receiver>
      <name>foo</name>
      <tls>
        <tcp-client-parameters>
          <remote-address>receiver.example.com</remote-address>
          <remote-port>443</remote-port>
        </tcp-client-parameters>
        <tls-client-parameters>
          <server-authentication>
            <ca-certs>
              <local-definition>
                <certificate>
                  <name>Server Cert Issuer #1</name>
                  <cert-data>base64encodedvalue==</cert-data>
                </certificate>
              </local-definition>
            </ca-certs>
          </server-authentication>
        </tls-client-parameters>
        <http-client-parameters>
          <client-identity>
            <basic>
              <user-id>my-name</user-id>
              <cleartext-password>my-password</cleartext-password>
            </basic>
          </client-identity>
          <path>/some/path</path>
        </http-client-parameters>
      </tls>
    </https-receiver>
  </https-receivers>
</example-module>
```

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Eric Voit, Henning Rogge, Martin Bjorklund, Reshad Rahman, and Rob Wilton.

In addition, the authors would also like to thank Quifang Ma for providing thoughtful comments as part of shepherd writeup.

Authors' Addresses

Mahesh Jethanandani
Kloud Services
Email: mjethanandani@gmail.com

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

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

K. Watsen
Watsen Networks

10 July 2020

A YANG Data Model for a Keystore
draft-ietf-netconf-keystore-19

Abstract

This document defines a YANG 1.1 module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

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

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

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

* "CCCC" --> the assigned RFC value for this draft

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

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

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

* Appendix A. Change Log

Status of This Memo

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

This Internet-Draft will expire on 11 January 2021.

Table of Contents

1. Introduction

This document defines a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

The "ietf-keystore" module defines many "grouping" statements intended for use by other modules that may import it. For instance, there are groupings that defined enabling a key to be either configured locally (within the defining data model) or be a reference to a key in the Keystore.

Special consideration has been given for systems that have cryptographic hardware, such as a Trusted Protection Module (TPM). These systems are unique in that the cryptographic hardware hides the secret key values. To support such hardware, symmetric keys may have the value "hidden-key" and asymmetric keys may have the value "hidden-private-key". While how such keys are created or destroyed is outside the scope of this document, the Keystore can contain entries for such keys, enabling them to be referenced by other configuration elements.

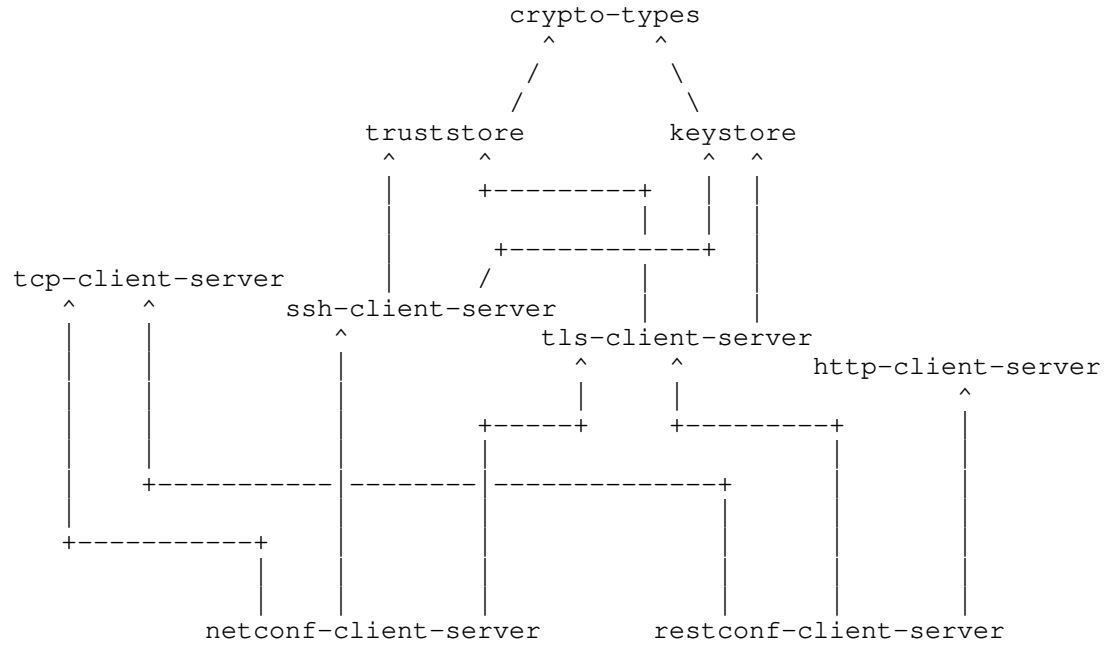
It is not required that a system has an operating system level keystore utility, with or without HSM backing, to implement this module. It is also possible that a system implementing the module to possess a multiplicity of operating system level keystore utilities and/or a multiplicity of HSMs.

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, keys and associated certificates installed during manufacturing (e.g., for an IDevID [Std-802.1AR-2009] certificate) are expected to appear in <operational> (see Section 3).

2. The "ietf-keystore" Module

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

2.1. Data Model Overview

2.1.1. Features

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

```
Features:
  +-- keystore-supported
  +-- local-definitions-supported
```

2.1.2. Typedefs

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

```
Typedefs:
  leafref
    +-- symmetric-key-ref
    +-- asymmetric-key-ref
```

Comments:

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

* The leafrefs refer to symmetric and asymmetric keys in the keystore. These typedefs are provided primarily as an aid to downstream modules that import the "ietf-keystore" module.

2.1.3. Groupings

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

```
Groupings:
  +-- encrypted-by-choice-grouping
  +-- asymmetric-key-certificate-ref-grouping
  +-- local-or-keystore-symmetric-key-grouping
  +-- local-or-keystore-asymmetric-key-grouping
  +-- local-or-keystore-asymmetric-key-with-certs-grouping
  +-- local-or-keystore-end-entity-cert-with-key-grouping
  +-- keystore-grouping
```

Each of these groupings are presented in the following subsections.

2.1.3.1. The "encrypted-by-choice-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "encrypted-by-choice-grouping" grouping:

```
   |  The grouping's name is intended to be parsed "(encrypted-by)-
   |  (choice-grouping)", not as "(encrypted)-(by-
   |  choice)-(grouping)".

  grouping encrypted-by-choice-grouping
    +-- (encrypted-by-choice)
       +--:(symmetric-key-ref)
       |  +-- symmetric-key-ref?
       |           -> /keystore/symmetric-keys/symmetric-key/name
       +--:(asymmetric-key-ref)
          +-- asymmetric-key-ref?
                  -> /keystore/asymmetric-keys/asymmetric-key/name
```

Comments:

* This grouping defines a "choice" statement with options to reference either a symmetric or an asymmetric key configured in the keystore.

2.1.3.2. The "asymmetric-key-certificate-ref-grouping" Grouping

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

```
  grouping asymmetric-key-certificate-ref-grouping
    +-- asymmetric-key?   ks:asymmetric-key-ref
    +-- certificate?      leafref
```

Comments:

* This grouping defines a reference to a certificate in two parts: the first being the name of the asymmetric key the certificate is associated with, and the second being the name of the certificate itself.

2.1.3.3. The "local-or-keystore-symmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-symmetric-key-grouping" grouping:

```
  grouping local-or-keystore-symmetric-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +---u ct:symmetric-key-grouping
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference?   ks:symmetric-key-ref
```

Comments:

* The "local-or-keystore-symmetric-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option as to if an symmetric key is defined locally or as a reference to a symmetric key in the keystore.

* 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 symmetric key in an alternate location.

* For the "local-definition" option, the defintion uses the "symmetric-key-grouping" grouping discussed in Section 2.1.3.2 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "symmetric-key-ref" discussed in Section 2.1.2.

2.1.3.4. The "local-or-keystore-asymmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-grouping" grouping:

```
  grouping local-or-keystore-asymmetric-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-grouping
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference?   ks:asymmetric-key-ref
```

Comments:

* The "local-or-keystore-asymmetric-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option as to if an asymmetric key is defined locally or as a reference to a asymmetric key in the keystore.

* 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 asymmetric key in an alternate location.

* For the "local-definition" option, the defintion uses the "asymmetric-key-pair-grouping" grouping discussed in Section 2.1.3.4 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

2.1.3.5. The "local-or-keystore-asymmetric-key-with-certs-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-with-certs-grouping" grouping:

```
  grouping local-or-keystore-asymmetric-key-with-certs-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-with-certs-grouping
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference?   ks:asymmetric-key-ref
```

Comments:

* The "local-or-keystore-asymmetric-key-with-certs-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option as to if an asymmetric key is defined locally or as a reference to a asymmetric key in the keystore.

* 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 asymmetric key in an alternate location.

* For the "local-definition" option, the defintion uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.3.10 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

2.1.3.6. The "local-or-keystore-end-entity-cert-with-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-end-entity-cert-with-key-grouping" grouping:

```
  grouping local-or-keystore-end-entity-cert-with-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-with-cert-grouping
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference
             +---u asymmetric-key-certificate-ref-grouping
```

Comments:

* The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option as to if an symmetric key is defined locally or as a reference to a symmetric key in the keystore.

* 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 symmetric key in an alternate location.

* For the "local-definition" option, the defintion uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.3.10 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" uses the "asymmetric-key-certificate-ref-grouping" grouping discussed in Section 2.1.3.2.

2.1.3.7. The "keystore-grouping" Grouping

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

```
  grouping keystore-grouping
    +-- asymmetric-keys
    |  +-- asymmetric-key* [name]
    |     +-- name?                                          string
    |     +---u ct:asymmetric-key-pair-with-certs-grouping
    +-- symmetric-keys
       +-- symmetric-key* [name]
          +-- name?                        string
          +---u ct:symmetric-key-grouping
```

Comments:

* The "keystore-grouping" grouping is defines a keystore instance as being composed of symmetric and asymmetric keys. The stucture for the symmetric and asymmetric keys is essentially the same, being a "list" inside a "container".

* For asymmetric keys, each "asymmetric-key" uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed Section 2.1.3.10 of [I-D.ietf-netconf-crypto-types].

* For symmetric keys, each "symmetric-key" uses the "symmetric-key-grouping" grouping discussed Section 2.1.3.2 of [I-D.ietf-netconf-crypto-types].

2.1.4. Protocol-accessible Nodes

The following diagram lists all the protocol-accessible nodes defined in the "ietf-keystore" module:

```
module: ietf-keystore
  +--rw keystore
     +--rw asymmetric-keys
     |  +--rw asymmetric-key* [name]
     |     +--rw name                                    string
     |     +--rw public-key-format                       identityref
     |     +--rw public-key                              binary
     |     +--rw private-key-format?                     identityref
     |     +--rw (private-key-type)
     |     |  +--:(cleartext-private-key)
     |     |  |  +--rw cleartext-private-key?            binary
     |     |  +--:(hidden-private-key)
     |     |  |  +--rw hidden-private-key?               empty
     |     |  +--:(encrypted-private-key)
     |     |     +--rw encrypted-private-key
     |     |        +--rw encrypted-by
     |     |        |  +--rw (encrypted-by-choice)
     |     |        |     +--:(symmetric-key-ref)
     |     |        |     |  +--rw symmetric-key-ref?    leafref
     |     |        |     +--:(asymmetric-key-ref)
     |     |        |        +--rw asymmetric-key-ref?   leafref
     |     |        +--rw encrypted-value    binary
     |     +--rw certificates
     |     |  +--rw certificate* [name]
     |     |     +--rw name                       string
     |     |     +--rw cert-data                  end-entity-cert-cms
     |     |     +---n certificate-expiration
     |     |        +-- expiration-date    yang:date-and-time
     |     +---x generate-certificate-signing-request
     |             {certificate-signing-request-generation}?
```

```
        |        +---w input
        |        |  +---w csr-info    ct:csr-info
        |        +--ro output
        |           +--ro certificate-signing-request    ct:csr
        +--rw symmetric-keys
           +--rw symmetric-key* [name]
              +--rw name                    string
              +--rw key-format?             identityref
              +--rw (key-type)
                 +--:(cleartext-key)
                 |  +--rw cleartext-key?   binary
                 +--:(hidden-key)
                 |  +--rw hidden-key?      empty
                 +--:(encrypted-key)
                    +--rw encrypted-key
                       +--rw encrypted-by
                       |  +--rw (encrypted-by-choice)
                       |     +--:(symmetric-key-ref)
                       |     |  +--rw symmetric-key-ref?    leafref
                       |     +--:(asymmetric-key-ref)
                       |        +--rw asymmetric-key-ref?   leafref
                       +--rw encrypted-value    binary
```

Comments:

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

* For the "ietf-keystore" module, the protcol-accessible nodes are an instance of the "keystore-grouping" discussed in Section 2.1.3.7 grouping. Note that, in this diagram, all the used groupings have been expanded, enabling the keystore's full structure to be seen.

* The reason for why "keystore-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of the keystore 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 Keystore Instance

The following example illustrates keys in <running>. Please see Section 3 for an example illustrating built-in values in <operational>.

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

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

   <symmetric-keys>
      <symmetric-key>
         <name>cleartext-symmetric-key</name>
         <key-format>ct:octet-string-key-format</key-format>
         <cleartext-key>base64encodedvalue==</cleartext-key>
      </symmetric-key>
      <symmetric-key>
         <name>hidden-symmetric-key</name>
         <hidden-key/>
      </symmetric-key>
      <symmetric-key>
         <name>encrypted-symmetric-key</name>
         <key-format>
            ct:encrypted-one-symmetric-key-format
         </key-format>
         <encrypted-key>
           <encrypted-by>
             <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-k\
ey-ref>
           </encrypted-by>
           <encrypted-value>base64encodedvalue==</encrypted-value>
         </encrypted-key>
      </symmetric-key>
   </symmetric-keys>

   <asymmetric-keys>
      <asymmetric-key>
         <name>ssh-rsa-key</name>
         <public-key-format>
            ct:ssh-public-key-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
            ct:rsa-private-key-format
         </private-key-format>
         <cleartext-private-key>base64encodedvalue==</cleartext-priv\
ate-key>
```

```
        </asymmetric-key>
        <asymmetric-key>
           <name>ssh-rsa-key-with-cert</name>
           <public-key-format>
              ct:subject-public-key-info-format
           </public-key-format>
           <public-key>base64encodedvalue==</public-key>
           <private-key-format>
              ct:rsa-private-key-format
           </private-key-format>
           <cleartext-private-key>base64encodedvalue==</cleartext-priv\
   ate-key>
           <certificates>
              <certificate>
                 <name>ex-rsa-cert2</name>
                 <cert-data>base64encodedvalue==</cert-data>
              </certificate>
           </certificates>
        </asymmetric-key>
        <asymmetric-key>
           <name>raw-private-key</name>
           <public-key-format>
              ct:subject-public-key-info-format
           </public-key-format>
           <public-key>base64encodedvalue==</public-key>
           <private-key-format>
              ct:rsa-private-key-format
           </private-key-format>
           <cleartext-private-key>base64encodedvalue==</cleartext-priv\
   ate-key>
        </asymmetric-key>
        <asymmetric-key>
           <name>rsa-asymmetric-key</name>
           <public-key-format>
              ct:subject-public-key-info-format
           </public-key-format>
           <public-key>base64encodedvalue==</public-key>
           <private-key-format>
              ct:rsa-private-key-format
           </private-key-format>
           <cleartext-private-key>base64encodedvalue==</cleartext-priv\
   ate-key>
           <certificates>
              <certificate>
                 <name>ex-rsa-cert</name>
                 <cert-data>base64encodedvalue==</cert-data>
              </certificate>
           </certificates>
```

```
          </asymmetric-key>
          <asymmetric-key>
             <name>ec-asymmetric-key</name>
             <public-key-format>
                ct:subject-public-key-info-format
             </public-key-format>
             <public-key>base64encodedvalue==</public-key>
             <private-key-format>
                ct:ec-private-key-format
             </private-key-format>
             <cleartext-private-key>base64encodedvalue==</cleartext-priv\
   ate-key>
             <certificates>
                <certificate>
                   <name>ex-ec-cert</name>
                   <cert-data>base64encodedvalue==</cert-data>
                </certificate>
             </certificates>
          </asymmetric-key>
          <asymmetric-key>
             <name>hidden-asymmetric-key</name>
             <public-key-format>
                ct:subject-public-key-info-format
             </public-key-format>
             <public-key>base64encodedvalue==</public-key>
             <hidden-private-key/>
             <certificates>
                <certificate>
                   <name>builtin-idevid-cert</name>
                   <cert-data>base64encodedvalue==</cert-data>
                </certificate>
                <certificate>
                   <name>my-ldevid-cert</name>
                   <cert-data>base64encodedvalue==</cert-data>
                </certificate>
             </certificates>
          </asymmetric-key>
          <asymmetric-key>
             <name>encrypted-asymmetric-key</name>
             <public-key-format>
                ct:subject-public-key-info-format
             </public-key-format>
             <public-key>base64encodedvalue==</public-key>
             <private-key-format>
                ct:encrypted-one-asymmetric-key-format
             </private-key-format>
             <encrypted-private-key>
               <encrypted-by>
```

```
              <symmetric-key-ref>encrypted-symmetric-key</symmetric-k\
ey-ref>
            </encrypted-by>
            <encrypted-value>base64encodedvalue==</encrypted-value>
          </encrypted-private-key>
        </asymmetric-key>
      </asymmetric-keys>
   </keystore>
```

2.2.2. A Certificate Expiration Notification

The following example illustrates a "certificate-expiration" notification for a certificate associated with a key configured in the keystore.

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

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    <asymmetric-keys>
      <asymmetric-key>
        <name>hidden-asymmetric-key</name>
        <certificates>
          <certificate>
            <name>my-ldevid-cert</name>
            <certificate-expiration>
              <expiration-date>2018-08-05T14:18:53-05:00</expiration\
-date>
            </certificate-expiration>
          </certificate>
        </certificates>
      </asymmetric-key>
    </asymmetric-keys>
  </keystore>
</notification>
```

2.2.3. The "Local or Keystore" Groupings

This section illustrates the various "local-or-keystore" groupings defined in the "ietf-keystore" module, specifically the "local-or-keystore-symmetric-key-grouping" (Section 2.1.3.3), "local-or-keystore-asymmetric-key-grouping" (Section 2.1.3.4), "local-or-keystore-asymmetric-key-with-certs-grouping" (Section 2.1.3.5), and "local-or-keystore-end-entity-cert-with-key-grouping" (Section 2.1.3.6) groupings.

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

```
module ex-keystore-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-keystore-usage";
  prefix "eku";

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  organization
   "Example Corporation";

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

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

  revision "2020-07-10" {
    description
     "Initial version";
    reference
     "RFC CCCC: A YANG Data Model for a Keystore";
  }

  container keystore-usage {
    description
      "An illustration of the various keystore groupings.";

    list symmetric-key {
      key name;
      leaf name {
        type string;
        description
          "An arbitrary name for this key.";
      }
      uses ks:local-or-keystore-symmetric-key-grouping;
      description
        "An symmetric key that may be configured locally or be a
         reference to a symmetric key in the keystore.";
    }
```

```
      list asymmetric-key {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-grouping;
        description
          "An asymmetric key, with no certs, that may be configured
           locally or be a reference to an asymmetric key in the
           keystore.  The intent is to reference just the asymmetric
           key, not any certificates that may also be associated
           with the asymmetric key.";
      }

      list asymmetric-key-with-certs {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-with-certs-grouping;
        description
          "An asymmetric key and its associated certs, that may be
           configured locally or be a reference to an asymmetric key
           (and its associated certs) in the keystore.";
      }

      list end-entity-cert-with-key {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
        description
          "An end-entity certificate, and its associated private key,
           that may be configured locally or be a reference to a
           specific certificate (and its associated private key) in
           the keystore.";
      }
    }

  }
```

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

```
module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                          string
     |  +--rw (local-or-keystore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw key-format?            identityref
     |     |     +--rw (key-type)
     |     |        +--:(cleartext-key)
     |     |        |  +--rw cleartext-key?   binary
     |     |        +--:(hidden-key)
     |     |        |  +--rw hidden-key?      empty
     |     |        +--:(encrypted-key)
     |     |           +--rw encrypted-key
     |     |              +--rw encrypted-by
     |     |              +--rw encrypted-value    binary
     |     +--:(keystore) {keystore-supported}?
     |        +--rw keystore-reference?   ks:symmetric-key-ref
     +--rw asymmetric-key* [name]
     |  +--rw name                          string
     |  +--rw (local-or-keystore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw public-key-format              identityref
     |     |     +--rw public-key                     binary
     |     |     +--rw private-key-format?            identityref
     |     |     +--rw (private-key-type)
     |     |        +--:(cleartext-private-key)
     |     |        |  +--rw cleartext-private-key?   binary
     |     |        +--:(hidden-private-key)
     |     |        |  +--rw hidden-private-key?      empty
     |     |        +--:(encrypted-private-key)
     |     |           +--rw encrypted-private-key
     |     |              +--rw encrypted-by
     |     |              +--rw encrypted-value    binary
     |     +--:(keystore) {keystore-supported}?
     |        +--rw keystore-reference?   ks:asymmetric-key-ref
     +--rw asymmetric-key-with-certs* [name]
     |  +--rw name                          string
     |  +--rw (local-or-keystore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw public-key-format
     |     |     |       identityref
     |     |     +--rw public-key                                binary
```

```
          |      |     +--rw private-key-format?
          |      |     |       identityref
          |      |     +--rw (private-key-type)
          |      |     |  +--:(cleartext-private-key)
          |      |     |  |  +--rw cleartext-private-key?            binary
          |      |     |  +--:(hidden-private-key)
          |      |     |  |  +--rw hidden-private-key?               empty
          |      |     |  +--:(encrypted-private-key)
          |      |     |     +--rw encrypted-private-key
          |      |     |        +--rw encrypted-by
          |      |     |        +--rw encrypted-value    binary
          |      |     +--rw certificates
          |      |     |  +--rw certificate* [name]
          |      |     |     +--rw name                        string
          |      |     |     +--rw cert-data
          |      |     |     |       end-entity-cert-cms
          |      |     |     +---n certificate-expiration
          |      |     |        +-- expiration-date    yang:date-and-time
          |      |     +---x generate-certificate-signing-request
          |      |     |       {certificate-signing-request-generation}?
          |      |     |  +---w input
          |      |     |  |  +---w csr-info    ct:csr-info
          |      |     |  +--ro output
          |      |     |     +--ro certificate-signing-request    ct:csr
          |      |     +--:(keystore) {keystore-supported}?
          |      |        +--rw keystore-reference?   ks:asymmetric-key-ref
          |     +--rw end-entity-cert-with-key* [name]
          |        +--rw name                          string
          |        +--rw (local-or-keystore)
          |           +--:(local) {local-definitions-supported}?
          |           |  +--rw local-definition
          |           |     +--rw public-key-format
          |           |     |       identityref
          |           |     +--rw public-key                                binary
          |           |     +--rw private-key-format?
          |           |     |       identityref
          |           |     +--rw (private-key-type)
          |           |     |  +--:(cleartext-private-key)
          |           |     |  |  +--rw cleartext-private-key?            binary
          |           |     |  +--:(hidden-private-key)
          |           |     |  |  +--rw hidden-private-key?               empty
          |           |     |  +--:(encrypted-private-key)
          |           |     |     +--rw encrypted-private-key
          |           |     |        +--rw encrypted-by
          |           |     |        +--rw encrypted-value    binary
          |           |     +--rw cert-data?
          |           |     |       end-entity-cert-cms
          |           |     +---n certificate-expiration
```

```
               |     |  +-- expiration-date    yang:date-and-time
               |     +---x generate-certificate-signing-request
               |             {certificate-signing-request-generation}?
               |        +---w input
               |        |  +---w csr-info    ct:csr-info
               |        +--ro output
               |           +--ro certificate-signing-request    ct:csr
               +--:(keystore) {keystore-supported}?
                  +--rw keystore-reference
                     +--rw asymmetric-key?   ks:asymmetric-key-ref
                     +--rw certificate?      leafref
```

The following example provides two equivalent instances of each grouping, the first being a reference to a keystore and the second being locally-defined. The instance having a reference to a keystore is consistent with the keystore defined in Section 2.2.1. The two instances are equivalent, as the locally-defined instance example contains the same values defined by the keystore instance referenced by its sibling example.

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

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

  <!-- The following two equivalent examples illustrate the -->
  <!-- "local-or-keystore-symmetric-key-grouping" grouping: -->

  <symmetric-key>
    <name>example 1a</name>
    <keystore-reference>cleartext-symmetric-key</keystore-reference>
  </symmetric-key>

  <symmetric-key>
    <name>example 1b</name>
    <local-definition>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-key>base64encodedvalue==</cleartext-key>
    </local-definition>
  </symmetric-key>


  <!-- The following two equivalent examples illustrate the  -->
  <!-- "local-or-keystore-asymmetric-key-grouping" grouping: -->

  <asymmetric-key>
    <name>example 2a</name>
```

```
      <keystore-reference>rsa-asymmetric-key</keystore-reference>
    </asymmetric-key>

    <asymmetric-key>
      <name>example 2b</name>
      <local-definition>
        <public-key-format>
          ct:subject-public-key-info-format
        </public-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>
          ct:rsa-private-key-format
        </private-key-format>
        <cleartext-private-key>base64encodedvalue==</cleartext-private\
  -key>
      </local-definition>
    </asymmetric-key>


    <!-- the following two equivalent examples illustrate        -->
    <!-- "local-or-keystore-asymmetric-key-with-certs-grouping": -->

    <asymmetric-key-with-certs>
      <name>example 3a</name>
      <keystore-reference>rsa-asymmetric-key</keystore-reference>
    </asymmetric-key-with-certs>

    <asymmetric-key-with-certs>
      <name>example 3b</name>
      <local-definition>
        <public-key-format>
            ct:subject-public-key-info-format
        </public-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>
          ct:rsa-private-key-format
        </private-key-format>
        <cleartext-private-key>base64encodedvalue==</cleartext-private\
  -key>
        <certificates>
          <certificate>
            <name>a locally-defined cert</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </certificates>
      </local-definition>
    </asymmetric-key-with-certs>
```

```
     <!-- The following two equivalent examples illustrate       -->
     <!-- "local-or-keystore-end-entity-cert-with-key-grouping": -->

     <end-entity-cert-with-key>
       <name>example 4a</name>
       <keystore-reference>
         <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
         <certificate>ex-rsa-cert</certificate>
       </keystore-reference>
     </end-entity-cert-with-key>

     <end-entity-cert-with-key>
       <name>example 4b</name>
       <local-definition>
         <public-key-format>
           ct:subject-public-key-info-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
           ct:rsa-private-key-format
         </private-key-format>
         <cleartext-private-key>base64encodedvalue==</cleartext-private\
   -key>
         <cert-data>base64encodedvalue==</cert-data>
       </local-definition>
     </end-entity-cert-with-key>

   </keystore-usage>
```

2.3. YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types].

```
   <CODE BEGINS> file "ietf-keystore@2020-07-10.yang"

   module ietf-keystore {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
     prefix ks;

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

     import ietf-crypto-types {
```

```
      prefix ct;
      reference
        "RFC AAAA: YANG Data Types and Groupings for Cryptography";
    }

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

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

    description
      "This module defines a Keystore to centralize management
       of security credentials.

       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 CCCC
       (https://www.rfc-editor.org/info/rfcCCCC); see the RFC
       itself for full legal notices.

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

    revision 2020-07-10 {
      description
        "Initial version";
      reference
        "RFC CCCC: A YANG Data Model for a Keystore";
    }

    /****************/
    /*   Features   */
    /****************/
```

```
  feature keystore-supported {
    description
      "The 'keystore-supported' feature indicates that the server
       supports the Keystore.";
  }

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

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

  typedef symmetric-key-ref {
    type leafref {
      path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key"
         + "/ks:name";
    }
    description
      "This typedef enables modules to easily define a reference
       to a symmetric key stored in the Keystore.";
  }

  typedef asymmetric-key-ref {
    type leafref {
      path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
         + "/ks:name";
    }
    description
      "This typedef enables modules to easily define a reference
       to an asymmetric key stored in the Keystore.";
  }

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

  grouping encrypted-by-choice-grouping {
    description
      "A grouping that defines a choice enabling references
       to other keys.";
    choice encrypted-by-choice {
      nacm:default-deny-write;
      mandatory true;
      description
```

```
          "A choice amongst other symmetric or asymmetric keys.";
        case symmetric-key-ref {
          leaf symmetric-key-ref {
            type leafref {
              path "/ks:keystore/ks:symmetric-keys/"
                   + "ks:symmetric-key/ks:name";
            }
            description
             "Identifies the symmetric key used to encrypt this key.";
          }
        }
        case asymmetric-key-ref {
          leaf asymmetric-key-ref {
            type leafref {
              path "/ks:keystore/ks:asymmetric-keys/"
                   + "ks:asymmetric-key/ks:name";
            }
            description
             "Identifies the asymmetric key used to encrypt this key.";
          }
        }
      }
    }

    grouping asymmetric-key-certificate-ref-grouping {
      description
        "This grouping defines a reference to a specific certificate
         associated with an asymmetric key stored in the Keystore.";
      leaf asymmetric-key {
        nacm:default-deny-write;
        type ks:asymmetric-key-ref;
        must '../certificate';
        description
          "A reference to an asymmetric key in the Keystore.";
      }
      leaf certificate {
        nacm:default-deny-write;
        type leafref {
          path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key[ks:"
               + "name = current()/../asymmetric-key]/ks:certificates"
               + "/ks:certificate/ks:name";
        }
        must '../asymmetric-key';
        description
          "A reference to a specific certificate of the
           asymmetric key in the Keystore.";
      }
    }
```

```
  // local-or-keystore-* groupings

  grouping local-or-keystore-symmetric-key-grouping {
    description
      "A grouping that expands to allow the symmetric key to be
       either stored locally, within the using data model, or be
       a reference to a symmetric key stored in the Keystore.";
    choice local-or-keystore {
      nacm:default-deny-write;
      mandatory true;
      description
        "A choice between an inlined definition and a definition
         that exists in the Keystore.";
      case local {
        if-feature "local-definitions-supported";
        container local-definition {
          description
            "Container to hold the local key definition.";
          uses ct:symmetric-key-grouping;
        }
      }
      case keystore {
        if-feature "keystore-supported";
        leaf keystore-reference {
          type ks:symmetric-key-ref;
          description
            "A reference to an symmetric key that exists in
             the Keystore.";
        }
      }
    }
  }

  grouping local-or-keystore-asymmetric-key-grouping {
    description
      "A grouping that expands to allow the asymmetric key to be
       either stored locally, within the using data model, or be
       a reference to an asymmetric key stored in the Keystore.";
    choice local-or-keystore {
      nacm:default-deny-write;
      mandatory true;
      case local {
        if-feature "local-definitions-supported";
        container local-definition {
          description
            "Container to hold the local key definition.";
          uses ct:asymmetric-key-pair-grouping;
        }
```

```
        }
        case keystore {
          if-feature "keystore-supported";
          leaf keystore-reference {
            type ks:asymmetric-key-ref;
            description
              "A reference to an asymmetric key that exists in
               the Keystore.  The intent is to reference just the
               asymmetric key without any regard for any certificates
               that may be associated with it.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the Keystore.";
      }
    }

    grouping local-or-keystore-asymmetric-key-with-certs-grouping {
      description
        "A grouping that expands to allow an asymmetric key and its
         associated certificates to be either stored locally, within
         the using data model, or be a reference to an asymmetric key
         (and its associated certificates) stored in the Keystore.";
      choice local-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses ct:asymmetric-key-pair-with-certs-grouping;
          }
        }
        case keystore {
          if-feature "keystore-supported";
          leaf keystore-reference {
            type ks:asymmetric-key-ref;
            description
              "A reference to an asymmetric-key (and all of its
               associated certificates) in the Keystore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the Keystore.";
      }
```

```
    }

    grouping local-or-keystore-end-entity-cert-with-key-grouping {
      description
        "A grouping that expands to allow an end-entity certificate
         (and its associated private key) to be either stored locally,
         within the using data model, or be a reference to a specific
         certificate in the Keystore.";
      choice local-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses ct:asymmetric-key-pair-with-cert-grouping;
          }
        }
        case keystore {
          if-feature "keystore-supported";
          container keystore-reference {
            uses asymmetric-key-certificate-ref-grouping;
            description
              "A reference to a specific certificate (and its
               associated private key) in the Keystore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the Keystore.";
      }
    }

    grouping keystore-grouping {
      description
        "Grouping definition enables use in other contexts.  If ever
         done, implementations SHOULD augment new 'case' statements
         into local-or-keystore 'choice' statements to supply leafrefs
         to the new location.";
      container asymmetric-keys {
        nacm:default-deny-write;
        description
          "A list of asymmetric keys.";
        list asymmetric-key {
          key "name";
          description
            "An asymmetric key.";
```

```
          leaf name {
            type string;
            description
              "An arbitrary name for the asymmetric key.";
          }
          uses ct:asymmetric-key-pair-with-certs-grouping;
        }
      }
      container symmetric-keys {
        nacm:default-deny-write;
        description
          "A list of symmetric keys.";
        list symmetric-key {
          key "name";
          description
            "A symmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the symmetric key.";
          }
          uses ct:symmetric-key-grouping;
        }
      }
    } // grouping keystore-grouping



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

    container keystore {
      description
        "The Keystore contains a list of symmetric keys and a list
         of asymmetric keys.";
      nacm:default-deny-write;
      uses keystore-grouping {
        augment "symmetric-keys/symmetric-key/key-type/encrypted-key/"
                + "encrypted-key/encrypted-by" {
          description
            "Augments in a choice statement enabling the encrypting
             key to be any other symmetric or asymmetric key in the
             keystore.";
          uses encrypted-by-choice-grouping;
        }
        augment "asymmetric-keys/asymmetric-key/private-key-type/"
                + "encrypted-private-key/encrypted-private-key/"
```

```
                + "encrypted-by" {
            description
              "Augments in a choice statement enabling the encrypting
               key to be any other symmetric or asymmetric key in the
               keystore.";
            uses encrypted-by-choice-grouping;
          }
        }
      }

   }

   <CODE ENDS>
```

3. Support for Built-in Keys

In some implementations, a server may support built-in keys. Built-in built-in keys MAY be set during the manufacturing process or be dynamically generated the first time the server is booted or a particular service (e.g., SSH) is enabled.

The key characteristic of the built-in keys is that they are provided by the system, as opposed to configuration. As such, they are present in <operational>. The example below illustrates what the keystore in <operational> might look like for a server in its factory default state.

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  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">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

In order for the built-in keys (and/or their associated built-in certificates) to be referenced by configuration, the referenced keys MUST first be copied into <running>. The keys SHOULD be copied into <running> using the same "key" values, so that the server can bind the references to the built-in entries.

Built-in "hidden" keys cannot be copied into other parts of the configuration because their private parts are hidden, and therefore impossible to replicate. Built-in "encrypted" keys MAY be copied into other parts of the configuration so long as they maintain their reference to the other built-in key that encrypted them.

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

A server MUST reject attempts to modify any aspect of built-in keys, with exception to associating additional certificates to a built-in key. That these keys are "configured" in <running> is an illusion, as they are strictly a read-only subset of that which must already exist in <operational>.

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

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <asymmetric-keys>
    <asymmetric-key>
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
        <certificate>
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

After the above configuration is applied, <operational> should appear as follows:

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  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">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
        <certificate or:origin="or:intended">
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

4. Encrypting Keys in Configuration

This section describes an approach that enables all the private keys on a server to be encrypted, such that traditional backup/restore procedures can be used without concern for keys being compromised when in transit.

4.1. Root Key

The cornerstone to this solution is the existence of a "root" key that can be used to encrypt all the other keys. The server MUST be able to use this key to decrypt the other keys in the configuration.

The root key SHOULD be a hidden key, i.e., one whose private data has no presence in <running> or <operational> (see "hidden-key" and "hidden-private-key" in "ietf-crypto-types" [I-D.ietf-netconf-crypto-types]). If the server implementation does not support hidden keys, then the private data part of key MUST be protected by access control with access granted only to an administrator with special access control rights (e.g., an organization's crypto officer). Given the long lifetime of built-in keys (see Section 3), built-in keys MUST be hidden.

A hidden root key MAY be either a symmetric key or an asymmetric key. If the hidden root key is symmetric, then the server MUST provide APIs enabling other keys (ideally generated by the server) to be encrypted. If the hidden root key is asymmetric, then the server SHOULD provide APIs enabling other keys to be both generated and encrypted by it, but MAY alternatively enable administrators with special access control rights to generate and encrypt the other keys themselves, using the hidden key's public part. For practical reasons, an unhidden root key SHOULD be asymmetric, so that its public part can be accessed by other administrators without concern.

4.2. Configuring Encrypting Keys

Each time a new key is to be configured, it SHOULD be encrypted by the root key.

In "ietf-crypto-types" [I-D.ietf-netconf-crypto-types], the format for an encrypted symmetric key is described by the "encrypted-one-symmetric-key-format" identity, while the format for an encrypted asymmetric key is described by the "encrypted-one-asymmetric-key-format" identity

Ideally, the server implementation provides an API to generate a symmetric or asymmetric key, and encrypt the generated key using another key known to the system (e.g., the root key). Thusly administrators can safely call this API to configure new keys.

In case the server implementation does not provide such an API, then the generating and encrypting steps MAY be performed outside the server, e.g., by an administrator with special access control rights.

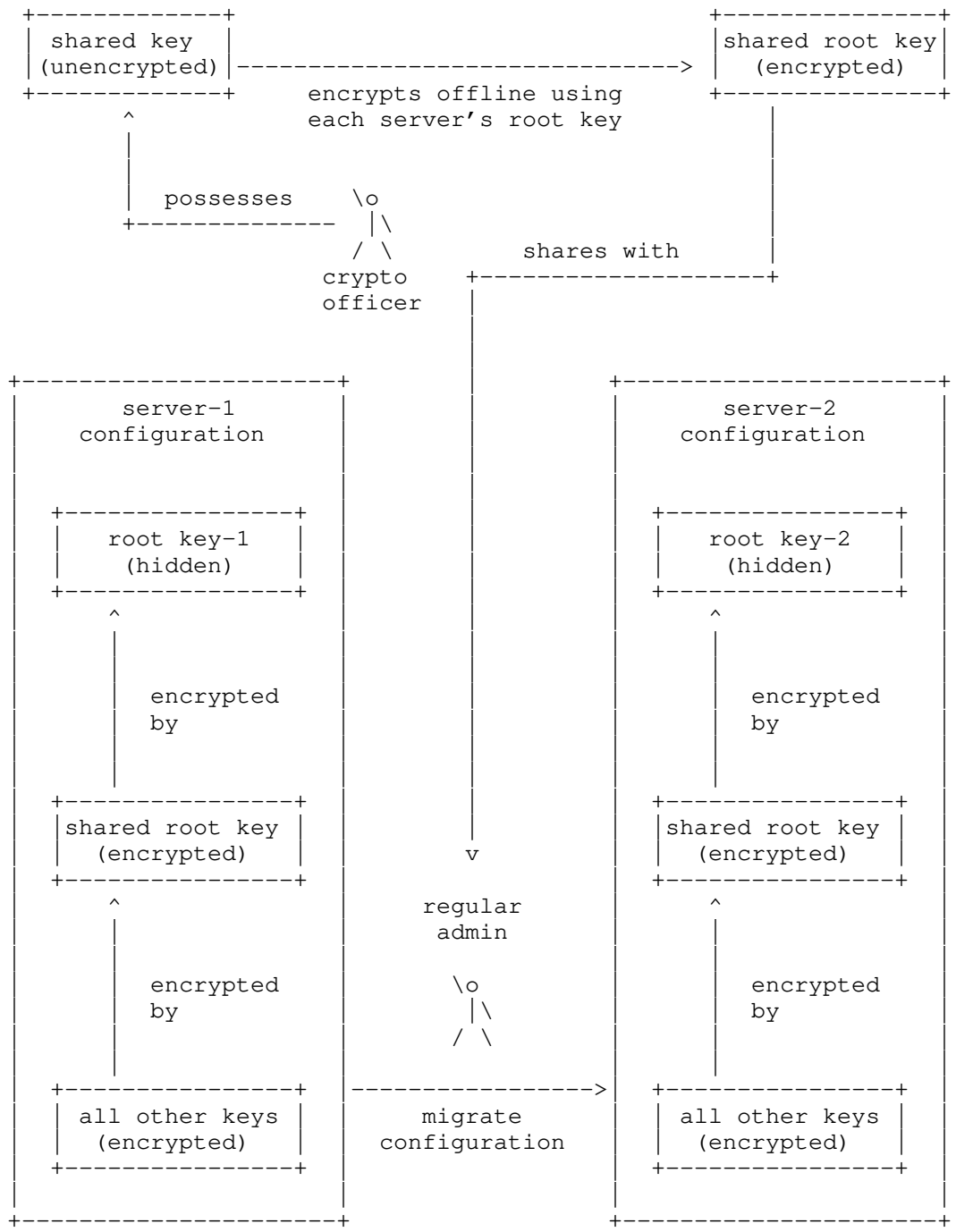
In either case, the encrypted key can be configured into the Keystore using either the "encrypted-key" (for symmetric keys) or the "encrypted-private-key" (for asymmetric keys) nodes. These two nodes contain both the encrypted value as well as a reference to the other key in the Keystore that it was encrypted by.

4.3. Migrating Configuration to Another Server

In the case a server's root key is used to encrypt other keys, migrating the configuration to another server may entail additional effort, assuming the second server has a different root key than the first server, in order for the second server to decrypt the other encrypted keys.

In some deployments, mechanisms outside the scope of this document may be used to migrate the root key from one server to another. That said, beware that the ability to do so typically entails having access to the first server but, in many RMA scenarios, the first server may no longer be operational.

Another option is to introduce a "shared root" key that acts as a portable intermediate root key. This shared root key would only need to be known to an organization's crypto officer. The shared root key SHOULD be encrypted offline by the crypto officer using each server's public key, which may be, e.g., in the server's IDevID certificate. The crypto officer can then safely handoff the encrypted shared key to other administrators responsible for server installations, including migrations. In order to migrate configuration from a first server, an administrator would need to make just a single modification to the configuration before loading it onto a second server, which is to replace the shared key's Keystore entry from the first server (an encrypted key), with the shared key encrypted by the second server's root key. The following diagram illustrates this idea:


shared key
(unencrypted)
encrypts offline using
each server's root key
shared root key
(encrypted)
possesses
crypto
officer
shares with
server-1
configuration
root key-1
(hidden)
encrypted
by
shared root key
(encrypted)
encrypted
by
all other keys
(encrypted)
regular
admin
migrate
configuration
server-2
configuration
root key-2
(hidden)
encrypted
by
shared root key
(encrypted)
encrypted
by
all other keys
(encrypted)

5. Security Considerations

5.1. Data at Rest

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

Security controls for the API (i.e., data in motion) are discussed in Section 5.2, but controls for the data at rest cannot be specified by the YANG module.

In order to satisfy the expectations of a "keystore", it is RECOMMENDED that implementations ensure that the keystore contents are encrypted when persisted to non-volatile memory.

5.2. The "ietf-keystore" YANG Module

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

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

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

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All of the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "keystore" instance, may be considered sensitive or vulnerable in some network environments.. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. 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-keystore
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

6.2. 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-keystore
namespace:    urn:ietf:params:xml:ns:yang:ietf-keystore
prefix:       ks
reference:    RFC CCCC
```

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[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>.

[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>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

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

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-

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

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

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

[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>.

[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>.

[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>.

[Std-802.1AR-2009]
Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

Appendix A. Change Log

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

A.1. 00 to 01

* Replaced the 'certificate-chain' structures with PKCS#7 structures. (Issue #1)
* Added 'private-key' as a configurable data node, and removed the 'generate-private-key' and 'load-private-key' actions. (Issue #2)
* Moved 'user-auth-credentials' to the ietf-ssh-client module. (Issues #4 and #5)

A.2. 01 to 02

* Added back 'generate-private-key' action.
* Removed 'RESTRICTED' enum from the 'private-key' leaf type.
* Fixed up a few description statements.

A.3. 02 to 03

* Changed draft's title.
* Added missing references.
* Collapsed sections and levels.
* Added RFC 8174 to Requirements Language Section.
* Renamed 'trusted-certificates' to 'pinned-certificates'.
* Changed 'public-key' from config false to config true.
* Switched 'host-key' from OneAsymmetricKey to definition from RFC 4253.

A.4. 03 to 04

* Added typedefs around leafrefs to common keystore paths
* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Removed Design Considerations section

* Moved key and certificate definitions from data tree to groupings

A.5. 04 to 05

* Removed trust anchors (now in their own draft)

* Added back global keystore structure

* Added groupings enabling keys to either be locally defined or a reference to the keystore.

A.6. 05 to 06

* Added feature "local-keys-supported"

* Added nacm:default-deny-all and nacm:default-deny-write

* Renamed generate-asymmetric-key to generate-hidden-key

* Added an install-hidden-key action

* Moved actions inside fo the "asymmetric-key" container

* Moved some groupings to draft-ietf-netconf-crypto-types

A.7. 06 to 07

* Removed a "require-instance false"

* Clarified some description statements

* Improved the keystore-usage examples

A.8. 07 to 08

* Added "local-definition" containers to avoid posibility of the action/notification statements being under a "case" statement.

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

A.9. 08 to 09

* Added a 'description' statement to the 'must' in the /keystore/asymmetric-key node explaining that the descendent values may exist in <operational> only, and that implementation MUST assert that the values are either configured or that they exist in <operational>.

* Copied above 'must' statement (and description) into the local-or-keystore-asymmetric-key-grouping, local-or-keystore-asymmetric-key-with-certs-grouping, and local-or-keystore-end-entity-cert-with-key-grouping statements.

A.10. 09 to 10

* Updated draft title to match new truststore draft title

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

* Renamed feature from 'local-keys-supported' to 'local-definitions-supported' (same name used in truststore)

* Removed the either-all-or-none 'must' expressions for the key's 3-tuple values (since the values are now 'mandatory true' in crypto-types)

* Example updated to reflect 'mandatory true' change in crypto-types draft

A.11. 10 to 11

* Replaced typedef asymmetric-key-certificate-ref with grouping asymmetric-key-certificate-ref-grouping.

* Added feature feature 'key-generation'.

* Cloned groupings symmetric-key-grouping, asymmetric-key-pair-grouping, asymmetric-key-pair-with-cert-grouping, and asymmetric-key-pair-with-certs-grouping from crypto-keys, augmenting into each new case statements for values that have been encrypted by other keys in the keystore. Refactored keystore model to use these groupings.

* Added new 'symmetric-keys' lists, as a sibling to the existing 'asymmetric-keys' list.

* Added RPCs (not actions) 'generate-symmetric-key' and 'generate-asymmetric-key' to *return* a (potentially encrypted) key.

A.12. 11 to 12

* Updated to reflect crypto-type's draft using enumerations over identities.

* Added examples for the 'generate-symmetric-key' and 'generate-asymmetric-key' RPCs.

* Updated the Introduction section.

A.13. 12 to 13

* Updated examples to incorporate new "key-format" identities.

* Made the two "generate-*-key" RPCs be "action" statements instead.

A.14. 13 to 14

* Updated YANG module and examples to incorporate the new iana-*-algorithm modules in the crypto-types draft..

A.15. 14 to 15

* Added new "Support for Built-in Keys" section.

* Added 'must' expressions asserting that the 'key-format' leaf whenever an encrypted key is specified.

* Added local-or-keystore-symmetric-key-grouping for PSK support.

A.16. 15 to 16

* Moved the generate key actions to ietf-crypt-types as RPCs, which are augmented by ietf-keystore to support encrypted keys. Examples updated accordingly.

* Added a SSH certificate-based key (RFC 6187) and a raw private key to the example instance document (partly so they could be referenced by examples in the SSH and TLS client/server drafts.

A.17. 16 to 17

* Removed augments to the "generate-symmetric-key" and "generate-asymmetric-key" groupings.

* Removed "generate-symmetric-key" and "generate-asymmetric-key" examples.

* Removed the "algorithm" nodes from remaining examples.

* Updated the "Support for Built-in Keys" section.

* Added new section "Encrypting Keys in Configuration".

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

A.18. 17 to 18

* Removed dangling/unnecessary ref to RFC 8342.

* r/MUST/SHOULD/ wrt strength of keys being configured over transports.

* Added an example for the "certificate-expiration" notification.

* Clarified that OS MAY have a multiplicity of underlying keystores and/or HSMs.

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

* Clarified the "Migrating Configuration to Another Server" section.

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

* Updated the Security Considerations section.

A.19. 18 to 19

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Benoit Claise, Bert Wijnen, Balazs Kovacs, David Lamparter, Eric Voit, Ladislav Lhotka, Liang Xia, Juergen Schoenwaelder, Mahesh Jethanandani, Martin Bjorklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Reshad Rahman, Sean Turner, and Tom Petch.

Author’s Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

A YANG Data Model for a Keystore and Keystore Operations
draft-ietf-netconf-keystore-35

Abstract

This document presents a YANG module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted or hidden. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

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

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

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

* CCCC --> 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:

* 2024-03-16 --> the publication date of this draft

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

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on 17 September 2024.

Copyright Notice

Table of Contents

1. Introduction

This document presents a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted or hidden (see [I-D.ietf-netconf-crypto-types]). Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

The "ietf-keystore" module defines many "grouping" statements intended for use by other modules that may import it. For instance, there are groupings that define enabling a key to be either configured inline (within the defining data model) or as a reference to a key in the central keystore.

Special consideration has been given for servers that have cryptographic hardware, such as a Trusted Platform Module (TPM). These servers are unique in that the cryptographic hardware hides the secret key values. Additionally, such hardware is commonly initialized when manufactured to protect a "built-in" asymmetric key for which its public half is conveyed in an identity certificate (e.g., an IDevID [Std-802.1AR-2018] certificate). Please see Section 3 to see how built-in keys are supported.

This document is intended to reflect existing practices that many server implementations support at the time of writing. To simplify implementation, advanced key formats may be selectively implemented.

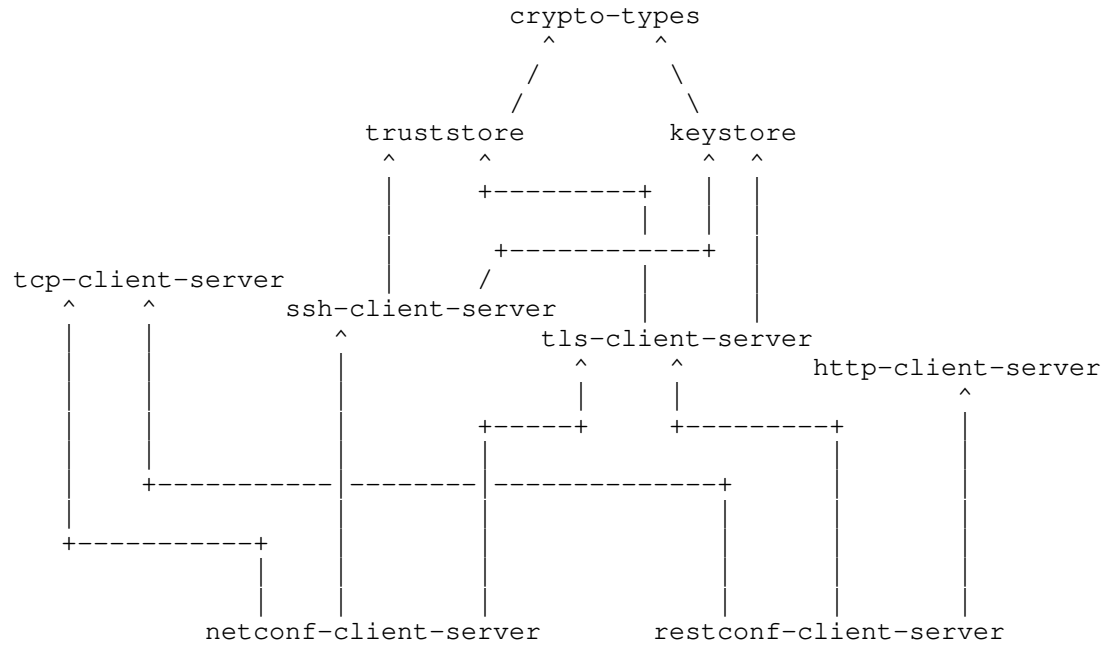
Implementations may utilize operating-system level keystore utilities (e.g., "Keychain Access" on MacOS) and/or cryptographic hardware (e.g., TPMs).

1.1. Relation to other RFCs

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

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



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

Table 1: Label 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. Terminology

The terms "client" and "server" are defined in [RFC6241] and are not redefined here.

The term "keystore" is defined in this document as a mechanism that intends to safeguard secrets.

The nomenclature "<running>" and "<operational>" are defined in [RFC8342].

The sentence fragments "augmented" and "augmented in" are used herein as the past tense verbified form of the "augment" statement defined in Section 7.17 of [RFC7950].

The term "key" may be used to mean one of three things in this document: 1) the YANG-defined "asymmetric-key" or "symmetric-key" node defined in this document, 2) the raw key data possessed by the aforementioned key nodes, and 3) the "key" of a YANG "list" statement. This document attempts to always qualify types '2' and '3' using, "raw key value" and "YANG list key" where needed. In all other cases, an unqualified "key" refers to a YANG-defined "asymmetric-key" or "symmetric-key" node.

1.4. Adherence to the NMDA

This document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, keys and associated certificates installed during manufacturing (e.g., for an IDevID certificate) are expected to appear in <operational> (see Section 3).

1.5. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). 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" to the word "keystore" to refer to the top-level instance of the "keystore-grouping", when the "central-keystore-supported" feature is enabled. Please be aware that consuming YANG modules MAY instantiate the "keystore-grouping" in other locations. All such other instances are not the "central" instance.

2. The "ietf-keystore" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-keystore". 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. The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-keystore" 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-keystore" module:

```
Features:
  +-- central-keystore-supported
  +-- inline-definitions-supported
  +-- asymmetric-keys
  +-- symmetric-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-keystore" module:

```
Typedefs:
  leafref
    +-- central-symmetric-key-ref
    +-- central-asymmetric-key-ref
```

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

Comments:

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

* The leafrefs refer to symmetric and asymmetric keys in the central keystore, when this module is implemented.

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

2.1.3. Groupings

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

* encrypted-by-grouping
* central-asymmetric-key-certificate-ref-grouping
* inline-or-keystore-symmetric-key-grouping
* inline-or-keystore-asymmetric-key-grouping
* inline-or-keystore-asymmetric-key-with-certs-grouping
* inline-or-keystore-end-entity-cert-with-key-grouping
* keystore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "encrypted-by-grouping" Grouping

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

```
  grouping encrypted-by-grouping:
    +-- (encrypted-by)
       +--:(central-symmetric-key-ref)
       |          {central-keystore-supported,symmetric-keys}?
       |  +-- symmetric-key-ref?    ks:central-symmetric-key-ref
       +--:(central-asymmetric-key-ref)
                  {central-keystore-supported,asymmetric-keys}?
          +-- asymmetric-key-ref?   ks:central-asymmetric-key-ref
```

Comments:

* This grouping defines a "choice" statement with options to reference either a symmetric or an asymmetric key configured in the keystore.

* This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations MUST augment in alternate "encrypted-by" references to the alternate locations.

2.1.3.2. The "central-asymmetric-key-certificate-ref-grouping" Grouping

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

```
  grouping central-asymmetric-key-certificate-ref-grouping:
    +-- asymmetric-key?   ks:central-asymmetric-key-ref
    |       {central-keystore-supported,asymmetric-keys}?
    +-- certificate?      leafref
```

Comments:

* This grouping defines a reference to a certificate in two parts: the first being the name of the asymmetric key the certificate is associated with, and the second being the name of the certificate itself.

* This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations can define an alternate grouping for references to the alternate locations.

2.1.3.3. The "inline-or-keystore-symmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "inline-or-keystore-symmetric-key-grouping" grouping:

```
  grouping inline-or-keystore-symmetric-key-grouping:
    +-- (inline-or-keystore)
       +--:(inline) {inline-definitions-supported}?
       |  +-- inline-definition
       |     +---u ct:symmetric-key-grouping
       +--:(central-keystore)
                {central-keystore-supported,symmetric-keys}?
          +-- central-keystore-reference?
                  ks:central-symmetric-key-ref
```

Comments:

* The "inline-or-keystore-symmetric-key-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for whether a symmetric key is defined inline or as a reference to a symmetric key in the keystore.

* 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 symmetric key in an alternate location.

* For the "inline-definition" option, the definition uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].

* For the "central-keystore" option, the "central-keystore-reference" is an instance of the "symmetric-key-ref" discussed in Section 2.1.2.

2.1.3.4. The "inline-or-keystore-asymmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "inline-or-keystore-asymmetric-key-grouping" grouping:

```
  grouping inline-or-keystore-asymmetric-key-grouping:
    +-- (inline-or-keystore)
       +--:(inline) {inline-definitions-supported}?
       |  +-- inline-definition
       |     +---u ct:asymmetric-key-pair-grouping
       +--:(central-keystore)
                {central-keystore-supported,asymmetric-keys}?
          +-- central-keystore-reference?
                  ks:central-asymmetric-key-ref
```

Comments:

* The "inline-or-keystore-asymmetric-key-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for whether an asymmetric key is defined inline or as a reference to an asymmetric key in the keystore.

* 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 an asymmetric key in an alternate location.

* For the "inline-definition" option, the definition uses the "asymmetric-key-pair-grouping" grouping discussed in Section 2.1.4.6 of [I-D.ietf-netconf-crypto-types].

* For the "central-keystore" option, the "central-keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

2.1.3.5. The "inline-or-keystore-asymmetric-key-with-certs-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "inline-or-keystore-asymmetric-key-with-certs-grouping" grouping:

```
  grouping inline-or-keystore-asymmetric-key-with-certs-grouping:
    +-- (inline-or-keystore)
       +--:(inline) {inline-definitions-supported}?
       |  +-- inline-definition
       |     +---u ct:asymmetric-key-pair-with-certs-grouping
       +--:(central-keystore)
                {central-keystore-supported,asymmetric-keys}?
          +-- central-keystore-reference?
                  ks:central-asymmetric-key-ref
```

Comments:

* The "inline-or-keystore-asymmetric-key-with-certs-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for whether an asymmetric key is defined inline or as a reference to an asymmetric key in the keystore.

* 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 an asymmetric key in an alternate location.

* For the "inline-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.12 of [I-D.ietf-netconf-crypto-types].

* For the "central-keystore" option, the "central-keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

2.1.3.6. The "inline-or-keystore-end-entity-cert-with-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "inline-or-keystore-end-entity-cert-with-key-grouping" grouping:

```
  grouping inline-or-keystore-end-entity-cert-with-key-grouping:
    +-- (inline-or-keystore)
       +--:(inline) {inline-definitions-supported}?
       |  +-- inline-definition
       |     +---u ct:asymmetric-key-pair-with-cert-grouping
       +--:(central-keystore)
                 {central-keystore-supported,asymmetric-keys}?
          +-- central-keystore-reference
             +---u central-asymmetric-key-certificate-ref-grouping
```

Comments:

* The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for whether a symmetric key is defined inline or as a reference to a symmetric key in the keystore.

* 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 symmetric key in an alternate location.

* For the "inline-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.12 of [I-D.ietf-netconf-crypto-types].

* For the "central-keystore" option, the "central-keystore-reference" uses the "central-asymmetric-key-certificate-ref-grouping" grouping discussed in Section 2.1.3.2.

2.1.3.7. The "keystore-grouping" Grouping

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

```
  grouping keystore-grouping:
    +-- asymmetric-keys {asymmetric-keys}?
    |  +-- asymmetric-key* [name]
    |     +-- name?                                          string
    |     +---u ct:asymmetric-key-pair-with-certs-grouping
    +-- symmetric-keys {symmetric-keys}?
       +-- symmetric-key* [name]
          +-- name?                          string
          +---u ct:symmetric-key-grouping
```

Comments:

* The "keystore-grouping" grouping defines a keystore instance as being composed of symmetric and asymmetric keys. The structure for the symmetric and asymmetric keys is essentially the same, being a "list" inside a "container".

* For asymmetric keys, each "asymmetric-key" uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.12 of [I-D.ietf-netconf-crypto-types].

* For symmetric keys, each "symmetric-key" uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].

2.1.4. Protocol-accessible Nodes

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

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

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

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

module: ietf-keystore
  +--rw keystore {central-keystore-supported}?
     +--rw asymmetric-keys {asymmetric-keys}?
     |  +--rw asymmetric-key* [name]
     |     +--rw name                              string
     |     +--rw public-key-format?                identityref
     |     +--rw public-key?                       binary
     |     +--rw private-key-format?               identityref
     |     +--rw (private-key-type)
     |     |  +--:(cleartext-private-key) {cleartext-private-keys}?
     |     |  |  +--rw cleartext-private-key?   binary
     |     |  +--:(hidden-private-key) {hidden-private-keys}?
     |     |  |  +--rw hidden-private-key?      empty
     |     |  +--:(encrypted-private-key) {encrypted-private-keys}?
     |     |     +--rw encrypted-private-key
     |     |        +--rw encrypted-by
     |     |        |  +--rw (encrypted-by)
     |     |        |     +--:(central-symmetric-key-ref)
     |     |        |     |        {central-keystore-supported,symme\
tric-keys}?
     |     |        |     |  +--rw symmetric-key-ref?
     |     |        |     |           ks:central-symmetric-key-ref
     |     |        |     +--:(central-asymmetric-key-ref)
     |     |        |              {central-keystore-supported,asymm\
etric-keys}?
     |     |        |        +--rw asymmetric-key-ref?
     |     |        |                 ks:central-asymmetric-key-ref
     |     |        +--rw encrypted-value-format    identityref
     |     |        +--rw encrypted-value           binary
     |     +--rw certificates
     |     |  +--rw certificate* [name]
     |     |     +--rw name                      string
     |     |     +--rw cert-data                 end-entity-cert-cms
     |     |     +---n certificate-expiration
     |     |             {certificate-expiration-notification}?
     |     |        +-- expiration-date    yang:date-and-time
     |     +---x generate-csr {csr-generation}?
     |        +---w input
     |        |  +---w csr-format    identityref
     |        |  +---w csr-info      csr-info
     |        +--ro output
```

```
        |           +--ro (csr-type)
        |              +--:(p10-csr)
        |                 +--ro p10-csr?   p10-csr
      +--rw symmetric-keys {symmetric-keys}?
         +--rw symmetric-key* [name]
            +--rw name                              string
            +--rw key-format?                       identityref
            +--rw (key-type)
               +--:(cleartext-symmetric-key)
               |  +--rw cleartext-symmetric-key?   binary
               |           {cleartext-symmetric-keys}?
               +--:(hidden-symmetric-key) {hidden-symmetric-keys}?
               |  +--rw hidden-symmetric-key?      empty
               +--:(encrypted-symmetric-key)
                        {encrypted-symmetric-keys}?
                  +--rw encrypted-symmetric-key
                     +--rw encrypted-by
                     |  +--rw (encrypted-by)
                     |     +--:(central-symmetric-key-ref)
                     |     |        {central-keystore-supported,symme\
tric-keys}?
                     |     |  +--rw symmetric-key-ref?
                     |     |           ks:central-symmetric-key-ref
                     |     +--:(central-asymmetric-key-ref)
                     |              {central-keystore-supported,asymm\
etric-keys}?
                     |        +--rw asymmetric-key-ref?
                     |                 ks:central-asymmetric-key-ref
                     +--rw encrypted-value-format    identityref
                     +--rw encrypted-value           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-keystore" module are instances of the "keystore-grouping" grouping discussed in Section 2.1.3.7.

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

* The "keystore-grouping" grouping is discussed in Section 2.1.3.7.

* The reason for why "keystore-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of the keystore 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 Keystore Instance

The following example illustrates keys in <running>. Please see Section 3 for an example illustrating built-in values in <operational>.

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

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

   <symmetric-keys>
      <symmetric-key>
         <name>cleartext-symmetric-key</name>
         <key-format>ct:octet-string-key-format</key-format>
         <cleartext-symmetric-key>BASE64VALUE=</cleartext-symmetric-\
key>
      </symmetric-key>
      <symmetric-key>
         <name>hidden-symmetric-key</name>
         <hidden-symmetric-key/>
      </symmetric-key>
      <symmetric-key>
         <name>encrypted-symmetric-key</name>
         <key-format>ct:one-symmetric-key-format</key-format>
         <encrypted-symmetric-key>
           <encrypted-by>
             <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-k\
ey-ref>
           </encrypted-by>
           <encrypted-value-format>ct:cms-enveloped-data-format</enc\
rypted-value-format>
           <encrypted-value>BASE64VALUE=</encrypted-value>
         </encrypted-symmetric-key>
      </symmetric-key>
   </symmetric-keys>
```

```
      <asymmetric-keys>
         <asymmetric-key>
            <name>ssh-rsa-key</name>
            <private-key-format>ct:rsa-private-key-format</private-key-\
   format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         </asymmetric-key>
         <asymmetric-key>
            <name>ssh-rsa-key-with-cert</name>
            <private-key-format>ct:rsa-private-key-format</private-key-\
   format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
            <certificates>
               <certificate>
                  <name>ex-rsa-cert2</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
         </asymmetric-key>
         <asymmetric-key>
            <name>raw-private-key</name>
            <private-key-format>ct:rsa-private-key-format</private-key-\
   format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         </asymmetric-key>
         <asymmetric-key>
            <name>rsa-asymmetric-key</name>
            <private-key-format>ct:rsa-private-key-format</private-key-\
   format>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
            <certificates>
               <certificate>
                  <name>ex-rsa-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
         </asymmetric-key>
         <asymmetric-key>
            <name>ec-asymmetric-key</name>
            <private-key-format>ct:ec-private-key-format</private-key-f\
   ormat>
            <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
            <certificates>
               <certificate>
                  <name>ex-ec-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
```

```
         </asymmetric-key>
         <asymmetric-key>
            <name>hidden-asymmetric-key</name>
            <public-key-format>ct:subject-public-key-info-format</publi\
   c-key-format>
            <public-key>BASE64VALUE=</public-key>
            <hidden-private-key/>
            <certificates>
               <certificate>
                  <name>builtin-idevid-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
               <certificate>
                  <name>my-ldevid-cert</name>
                  <cert-data>BASE64VALUE=</cert-data>
               </certificate>
            </certificates>
         </asymmetric-key>
         <asymmetric-key>
            <name>encrypted-asymmetric-key</name>
            <private-key-format>ct:one-asymmetric-key-format</private-k\
   ey-format>
            <encrypted-private-key>
              <encrypted-by>
                <symmetric-key-ref>encrypted-symmetric-key</symmetric-k\
   ey-ref>
              </encrypted-by>
              <encrypted-value-format>ct:cms-encrypted-data-format</enc\
   rypted-value-format>
              <encrypted-value>BASE64VALUE=</encrypted-value>
            </encrypted-private-key>
         </asymmetric-key>
      </asymmetric-keys>
   </keystore>
```

2.2.2. A Certificate Expiration Notification

The following example illustrates a "certificate-expiration" notification for a certificate associated with an asymmetric key configured in the keystore.

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

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
       <asymmetric-keys>
         <asymmetric-key>
           <name>hidden-asymmetric-key</name>
           <certificates>
             <certificate>
               <name>my-ldevid-cert</name>
               <certificate-expiration>
                 <expiration-date>2018-08-05T14:18:53-05:00</expiration\
   -date>
               </certificate-expiration>
             </certificate>
           </certificates>
         </asymmetric-key>
       </asymmetric-keys>
     </keystore>
   </notification>
```

2.2.3. The "Local or Keystore" Groupings

This section illustrates the various "inline-or-keystore" groupings defined in the "ietf-keystore" module, specifically the "inline-or-keystore-symmetric-key-grouping" (Section 2.1.3.3), "inline-or-keystore-asymmetric-key-grouping" (Section 2.1.3.4), "inline-or-keystore-asymmetric-key-with-certs-grouping" (Section 2.1.3.5), and "inline-or-keystore-end-entity-cert-with-key-grouping" (Section 2.1.3.6) groupings.

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

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

2.2.3.1. Tree Diagrams for the "ex-keystore-usage" Module

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

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

   module: ex-keystore-usage
     +--rw keystore-usage
        +--rw symmetric-key* [name]
        |  +--rw name                                                string
        |  +---u ks:inline-or-keystore-symmetric-key-grouping
        +--rw asymmetric-key* [name]
        |  +--rw name                                                 string
        |  +---u ks:inline-or-keystore-asymmetric-key-grouping
        +--rw asymmetric-key-with-certs* [name]
        |  +--rw name
        |  |       string
        |  +---u ks:inline-or-keystore-asymmetric-key-with-certs-groupi\
   ng
        +--rw end-entity-cert-with-key* [name]
           +--rw name
           |       string
           +---u ks:inline-or-keystore-end-entity-cert-with-key-grouping

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

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

   module: ex-keystore-usage
     +--rw keystore-usage
        +--rw symmetric-key* [name]
        |  +--rw name                                  string
        |  +--rw (inline-or-keystore)
        |     +--:(inline) {inline-definitions-supported}?
        |     |  +--rw inline-definition
        |     |     +--rw key-format?                      identityref
        |     |     +--rw (key-type)
        |     |        +--:(cleartext-symmetric-key)
        |     |        |  +--rw cleartext-symmetric-key?   binary
        |     |        |          {cleartext-symmetric-keys}?
        |     |        +--:(hidden-symmetric-key)
        |     |        |       {hidden-symmetric-keys}?
        |     |        |  +--rw hidden-symmetric-key?      empty
        |     |        +--:(encrypted-symmetric-key)
        |     |                {encrypted-symmetric-keys}?
        |     |           +--rw encrypted-symmetric-key
        |     |              +--rw encrypted-by
        |     |              +--rw encrypted-value-format    identityref
        |     |              +--rw encrypted-value           binary
        |     +--:(central-keystore)
```

```
        |              {central-keystore-supported,symmetric-keys}?
        |         +--rw central-keystore-reference?
        |                 ks:central-symmetric-key-ref
        +--rw asymmetric-key* [name]
        |  +--rw name                                  string
        |  +--rw (inline-or-keystore)
        |     +--:(inline) {inline-definitions-supported}?
        |     |  +--rw inline-definition
        |     |     +--rw public-key-format?             identityref
        |     |     +--rw public-key?                    binary
        |     |     +--rw private-key-format?            identityref
        |     |     +--rw (private-key-type)
        |     |        +--:(cleartext-private-key)
        |     |        |         {cleartext-private-keys}?
        |     |        |  +--rw cleartext-private-key?   binary
        |     |        +--:(hidden-private-key) {hidden-private-keys}?
        |     |        |  +--rw hidden-private-key?      empty
        |     |        +--:(encrypted-private-key)
        |     |                 {encrypted-private-keys}?
        |     |           +--rw encrypted-private-key
        |     |              +--rw encrypted-by
        |     |              +--rw encrypted-value-format    identityref
        |     |              +--rw encrypted-value           binary
        |     +--:(central-keystore)
        |              {central-keystore-supported,asymmetric-keys}?
        |        +--rw central-keystore-reference?
        |                 ks:central-asymmetric-key-ref
        +--rw asymmetric-key-with-certs* [name]
        |  +--rw name                                  string
        |  +--rw (inline-or-keystore)
        |     +--:(inline) {inline-definitions-supported}?
        |     |  +--rw inline-definition
        |     |     +--rw public-key-format?             identityref
        |     |     +--rw public-key?                    binary
        |     |     +--rw private-key-format?            identityref
        |     |     +--rw (private-key-type)
        |     |     |  +--:(cleartext-private-key)
        |     |     |  |         {cleartext-private-keys}?
        |     |     |  |  +--rw cleartext-private-key?   binary
        |     |     |  +--:(hidden-private-key) {hidden-private-keys}?
        |     |     |  |  +--rw hidden-private-key?      empty
        |     |     |  +--:(encrypted-private-key)
        |     |     |           {encrypted-private-keys}?
        |     |     |     +--rw encrypted-private-key
        |     |     |        +--rw encrypted-by
        |     |     |        +--rw encrypted-value-format    identityref
        |     |     |        +--rw encrypted-value           binary
        |     |     +--rw certificates
```

```
|      |      |  +--rw certificate* [name]
|      |      |     +--rw name                        string
|      |      |     +--rw cert-data
|      |      |     |       end-entity-cert-cms
|      |      |     +---n certificate-expiration
|      |      |             {certificate-expiration-notification}?
|      |      |        +-- expiration-date    yang:date-and-time
|      |      +---x generate-csr {csr-generation}?
|      |         +---w input
|      |         |  +---w csr-format    identityref
|      |         |  +---w csr-info      csr-info
|      |         +--ro output
|      |            +--ro (csr-type)
|      |               +--:(p10-csr)
|      |                  +--ro p10-csr?   p10-csr
|      +--:(central-keystore)
|               {central-keystore-supported,asymmetric-keys}?
|         +--rw central-keystore-reference?
|                 ks:central-asymmetric-key-ref
|
+--rw end-entity-cert-with-key* [name]
   +--rw name                                  string
   +--rw (inline-or-keystore)
      +--:(inline) {inline-definitions-supported}?
      |  +--rw inline-definition
      |     +--rw public-key-format?             identityref
      |     +--rw public-key?                    binary
      |     +--rw private-key-format?            identityref
      |     +--rw (private-key-type)
      |     |  +--:(cleartext-private-key)
      |     |  |        {cleartext-private-keys}?
      |     |  |  +--rw cleartext-private-key?   binary
      |     |  +--:(hidden-private-key) {hidden-private-keys}?
      |     |  |  +--rw hidden-private-key?      empty
      |     |  +--:(encrypted-private-key)
      |     |           {encrypted-private-keys}?
      |     |     +--rw encrypted-private-key
      |     |        +--rw encrypted-by
      |     |        +--rw encrypted-value-format    identityref
      |     |        +--rw encrypted-value           binary
      |     +--rw cert-data?
      |     |       end-entity-cert-cms
      |     +---n certificate-expiration
      |     |       {certificate-expiration-notification}?
      |     |  +-- expiration-date    yang:date-and-time
      |     +---x generate-csr {csr-generation}?
      |        +---w input
      |        |  +---w csr-format    identityref
      |        |  +---w csr-info      csr-info
```

```
               |        +--ro output
               |           +--ro (csr-type)
               |              +--:(p10-csr)
               |                 +--ro p10-csr?   p10-csr
               +--:(central-keystore)
                        {central-keystore-supported,asymmetric-keys}?
                  +--rw central-keystore-reference
                     +--rw asymmetric-key?
                     |       ks:central-asymmetric-key-ref
                     |       {central-keystore-supported,asymmetric-keys\
   }?
                     +--rw certificate?      leafref
```

2.2.3.2. Example Usage for the "ex-keystore-usage" Module

The following example provides two equivalent instances of each grouping, the first being a reference to a keystore and the second being inlined. The instance having a reference to a keystore is consistent with the keystore defined in Section 2.2.1. The two instances are equivalent, as the inlined instance example contains the same values defined by the keystore instance referenced by its sibling example.

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

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

  <!-- The following two equivalent examples illustrate the -->
  <!-- "inline-or-keystore-symmetric-key-grouping" grouping: -->

  <symmetric-key>
    <name>example 1a</name>
    <central-keystore-reference>cleartext-symmetric-key</central-key\
store-reference>
  </symmetric-key>

  <symmetric-key>
    <name>example 1b</name>
    <inline-definition>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-symmetric-key>BASE64VALUE=</cleartext-symmetric-key>
    </inline-definition>
  </symmetric-key>


  <!-- The following two equivalent examples illustrate the  -->
```

```
    <!-- "inline-or-keystore-asymmetric-key-grouping" grouping: -->

    <asymmetric-key>
      <name>example 2a</name>
      <central-keystore-reference>rsa-asymmetric-key</central-keystore\
  -reference>
    </asymmetric-key>

    <asymmetric-key>
      <name>example 2b</name>
      <inline-definition>
        <public-key-format>ct:subject-public-key-info-format</public-k\
  ey-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
  mat>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      </inline-definition>
    </asymmetric-key>


    <!-- the following two equivalent examples illustrate        -->
    <!-- "inline-or-keystore-asymmetric-key-with-certs-grouping": -->

    <asymmetric-key-with-certs>
      <name>example 3a</name>
      <central-keystore-reference>rsa-asymmetric-key</central-keystore\
  -reference>
    </asymmetric-key-with-certs>

    <asymmetric-key-with-certs>
      <name>example 3b</name>
      <inline-definition>
        <public-key-format>ct:subject-public-key-info-format</public-k\
  ey-format>
        <public-key>BASE64VALUE=</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
  mat>
        <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
        <certificates>
          <certificate>
            <name>a locally-defined cert</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </certificates>
      </inline-definition>
    </asymmetric-key-with-certs>
```

```
     <!-- The following two equivalent examples illustrate       -->
     <!-- "inline-or-keystore-end-entity-cert-with-key-grouping": -->

     <end-entity-cert-with-key>
       <name>example 4a</name>
       <central-keystore-reference>
         <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
         <certificate>ex-rsa-cert</certificate>
       </central-keystore-reference>
     </end-entity-cert-with-key>

     <end-entity-cert-with-key>
       <name>example 4b</name>
       <inline-definition>
         <public-key-format>ct:subject-public-key-info-format</public-k\
   ey-format>
         <public-key>BASE64VALUE=</public-key>
         <private-key-format>ct:rsa-private-key-format</private-key-for\
   mat>
         <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         <cert-data>BASE64VALUE=</cert-data>
       </inline-definition>
     </end-entity-cert-with-key>

   </keystore-usage>
```

2.2.3.3. The "ex-keystore-usage" YANG Module

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

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

     import ietf-keystore {
       prefix ks;
       reference
         "RFC CCCC: A YANG Data Model for a Keystore";
     }

     organization
       "Example Corporation";

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

     description
```

```
      "This example module illustrates notable groupings defined
       in the 'ietf-keystore' module.";

    revision 2024-03-16 {
      description
        "Initial version";
      reference
        "RFC CCCC: A YANG Data Model for a Keystore";
    }

    container keystore-usage {
      description
        "An illustration of the various keystore groupings.";
      list symmetric-key {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:inline-or-keystore-symmetric-key-grouping;
        description
          "An symmetric key that may be configured locally or be a
           reference to a symmetric key in the keystore.";
      }
      list asymmetric-key {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:inline-or-keystore-asymmetric-key-grouping;
        description
          "An asymmetric key, with no certs, that may be configured
           locally or be a reference to an asymmetric key in the
           keystore.  The intent is to reference just the asymmetric
           key, not any certificates that may also be associated
           with the asymmetric key.";
      }
      list asymmetric-key-with-certs {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:inline-or-keystore-asymmetric-key-with-certs-grouping;
```

```
        description
          "An asymmetric key and its associated certs, that may be
           configured locally or be a reference to an asymmetric key
           (and its associated certs) in the keystore.";
      }
      list end-entity-cert-with-key {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:inline-or-keystore-end-entity-cert-with-key-grouping;
        description
          "An end-entity certificate and its associated asymmetric
           key, that may be configured locally or be a reference
           to another certificate (and its associated asymmetric
           key) in the keystore.";
      }
    }
  }
```

2.3. YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types].

<CODE BEGINS> file "ietf-keystore@2024-03-16.yang"

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

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

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

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

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

  description
    "This module defines a 'keystore' to centralize management
     of security credentials.

     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 CCCC
     (https://www.rfc-editor.org/info/rfcCCCC); 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 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

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

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

```
  feature inline-definitions-supported {
    description
      "The 'inline-definitions-supported' feature indicates that
       the server supports locally-defined keys.";
  }

  feature asymmetric-keys {
    description
      "The 'asymmetric-keys' feature indicates that the server
       implements the /keystore/asymmetric-keys subtree.";

  }

  feature symmetric-keys {
    description
      "The 'symmetric-keys' feature indicates that the server
       implements the /keystore/symmetric-keys subtree.";
  }

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

  typedef central-symmetric-key-ref {
    type leafref {
      path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key"
         + "/ks:name";
    }
    description
      "This typedef enables modules to easily define a reference
       to a symmetric key stored in the central keystore.";
  }

  typedef central-asymmetric-key-ref {
    type leafref {
      path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
         + "/ks:name";
    }
    description
      "This typedef enables modules to easily define a reference
       to an asymmetric key stored in the central keystore.";
  }

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

  grouping encrypted-by-grouping {
```

```
      description
        "A grouping that defines a 'choice' statement that can be
         augmented into the 'encrypted-by' node, present in the
         'symmetric-key-grouping' and 'asymmetric-key-pair-grouping'
         groupings defined in RFC AAAA, enabling references to keys
         in the central keystore.";
      choice encrypted-by {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice amongst other symmetric or asymmetric keys.";
        case central-symmetric-key-ref {
          if-feature "central-keystore-supported";
          if-feature "symmetric-keys";
          leaf symmetric-key-ref {
            type ks:central-symmetric-key-ref;
            description
              "Identifies the symmetric key used to encrypt the
               associated key.";
          }
        }
        case central-asymmetric-key-ref {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          leaf asymmetric-key-ref {
            type ks:central-asymmetric-key-ref;
            description
              "Identifies the asymmetric key whose public key
               encrypted the associated key.";
          }
        }
      }
    }

    // *-ref groupings

    grouping central-asymmetric-key-certificate-ref-grouping {
      description
        "Grouping for the reference to a certificate associated
         with an asymmetric key stored in the central keystore.";
      leaf asymmetric-key {
        nacm:default-deny-write;
        if-feature "central-keystore-supported";
        if-feature "asymmetric-keys";
        type ks:central-asymmetric-key-ref;
        must '../certificate';
        description
          "A reference to an asymmetric key in the keystore.";
```

```
      }
      leaf certificate {
        nacm:default-deny-write;
        type leafref {
          path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
             + "[ks:name = current()/../asymmetric-key]/"
             + "ks:certificates/ks:certificate/ks:name";
        }
        must '../asymmetric-key';
        description
          "A reference to a specific certificate of the
           asymmetric key in the keystore.";
      }
    }

    // inline-or-keystore-* groupings

    grouping inline-or-keystore-symmetric-key-grouping {
      description
        "A grouping for the configuration of a symmetric key.  The
         symmetric key may be defined inline or as a reference to
         a symmetric key stored in the central keystore.

         Servers that wish to define alternate keystore locations
         SHOULD augment in custom 'case' statements enabling
         references to those alternate keystore locations.";
      choice inline-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "Container to hold the local key definition.";
            uses ct:symmetric-key-grouping;
          }
        }
        case central-keystore {
          if-feature "central-keystore-supported";
          if-feature "symmetric-keys";
          leaf central-keystore-reference {
            type ks:central-symmetric-key-ref;
            description
              "A reference to an symmetric key that exists in
               the central keystore.";
```

```
          }
        }
      }
    }

    grouping inline-or-keystore-asymmetric-key-grouping {
      description
        "A grouping for the configuration of an asymmetric key. The
         asymmetric key may be defined inline or as a reference to
         an asymmetric key stored in the central keystore.

         Servers that wish to define alternate keystore locations
         SHOULD augment in custom 'case' statements enabling
         references to those alternate keystore locations.";
      choice inline-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "Container to hold the local key definition.";
            uses ct:asymmetric-key-pair-grouping;
          }
        }
        case central-keystore {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          leaf central-keystore-reference {
            type ks:central-asymmetric-key-ref;
            description
              "A reference to an asymmetric key that exists in
               the central keystore.  The intent is to reference
               just the asymmetric key without any regard for
               any certificates that may be associated with it.";
          }
        }
      }
    }

    grouping inline-or-keystore-asymmetric-key-with-certs-grouping {
      description
        "A grouping for the configuration of an asymmetric key and
         its associated certificates.  The asymmetric key and its
         associated certificates may be defined inline or as a
```

```
          reference to an asymmetric key (and its associated
          certificates) in the central keystore.

          Servers that wish to define alternate keystore locations
          SHOULD augment in custom 'case' statements enabling
          references to those alternate keystore locations.";
        choice inline-or-keystore {
          nacm:default-deny-write;
          mandatory true;
          description
            "A choice between an inlined definition and a definition
             that exists in the keystore.";
          case inline {
            if-feature "inline-definitions-supported";
            container inline-definition {
              description
                "Container to hold the local key definition.";
              uses ct:asymmetric-key-pair-with-certs-grouping;
            }
          }
          case central-keystore {
            if-feature "central-keystore-supported";
            if-feature "asymmetric-keys";
            leaf central-keystore-reference {
              type ks:central-asymmetric-key-ref;
              description
                "A reference to an asymmetric-key (and all of its
                 associated certificates) in the keystore, when
                 this module is implemented.";
            }
          }
        }
      }

      grouping inline-or-keystore-end-entity-cert-with-key-grouping {
        description
          "A grouping for the configuration of an asymmetric key and
           its associated end-entity certificate.  The asymmetric key
           and its associated end-entity certificate may be defined
           inline or as a reference to an asymmetric key (and its
           associated end-entity certificate) in the central keystore.

           Servers that wish to define alternate keystore locations
           SHOULD augment in custom 'case' statements enabling
           references to those alternate keystore locations.";
        choice inline-or-keystore {
          nacm:default-deny-write;
          mandatory true;
```

```
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "Container to hold the local key definition.";
            uses ct:asymmetric-key-pair-with-cert-grouping;
          }
        }
        case central-keystore {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          container central-keystore-reference {
            uses central-asymmetric-key-certificate-ref-grouping;
            description
              "A reference to a specific certificate associated with
               an asymmetric key stored in the central keystore.";
          }
        }
      }
    }

    // the keystore grouping

    grouping keystore-grouping {
      description
        "Grouping definition enables use in other contexts.  If ever
         done, implementations MUST augment new 'case' statements
         into the various inline-or-keystore 'choice' statements to
         supply leafrefs to the model-specific location(s).";
      container asymmetric-keys {
        nacm:default-deny-write;
        if-feature "asymmetric-keys";
        description
          "A list of asymmetric keys.";
        list asymmetric-key {
          key "name";
          description
            "An asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the asymmetric key.";
          }
          uses ct:asymmetric-key-pair-with-certs-grouping;
        }
```

```
      }
      container symmetric-keys {
        nacm:default-deny-write;
        if-feature "symmetric-keys";
        description
          "A list of symmetric keys.";
        list symmetric-key {
          key "name";
          description
            "A symmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the symmetric key.";
          }
          uses ct:symmetric-key-grouping;
        }
      }
    }

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

    container keystore {
      if-feature central-keystore-supported;
      description
        "A central keystore containing a list of symmetric keys and
         a list of asymmetric keys.";
      nacm:default-deny-write;
      uses keystore-grouping {
        augment "symmetric-keys/symmetric-key/key-type/encrypted-"
              + "symmetric-key/encrypted-symmetric-key/encrypted-by" {
          description
            "Augments in a choice statement enabling the encrypting
             key to be any other symmetric or asymmetric key in the
             central keystore.";
          uses encrypted-by-grouping;
        }
        augment "asymmetric-keys/asymmetric-key/private-key-type/"
              + "encrypted-private-key/encrypted-private-key/"
              + "encrypted-by" {
          description
            "Augments in a choice statement enabling the encrypting
             key to be any other symmetric or asymmetric key in the
             central keystore.";
          uses encrypted-by-grouping;
        }
```

```
      }
    }
  }
```

<CODE ENDS>

3. Support for Built-in Keys

In some implementations, a server may support keys built into the server. Built-in keys MAY be set during the manufacturing process or be dynamically generated the first time the server is booted or a particular service (e.g., SSH) is enabled.

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

The primary characteristic of the built-in keys 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> [I-D.ietf-netmod-system-config], if implemented.

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

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

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  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">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

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

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

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <asymmetric-keys>
    <asymmetric-key>
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

After the above configuration is applied, <operational> should appear as follows:

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

   <keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
     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">
     <asymmetric-keys>
       <asymmetric-key or:origin="or:system">
         <name>Manufacturer-Generated Hidden Key</name>
         <public-key-format>ct:subject-public-key-info-format</public-k\
   ey-format>
         <public-key>BASE64VALUE=</public-key>
         <hidden-private-key/>
         <certificates>
           <certificate>
             <name>Manufacturer-Generated IDevID Cert</name>
             <cert-data>BASE64VALUE=</cert-data>
           </certificate>
           <certificate or:origin="or:intended">
             <name>Deployment-Specific LDevID Cert</name>
             <cert-data>BASE64VALUE=</cert-data>
           </certificate>
         </certificates>
       </asymmetric-key>
     </asymmetric-keys>
   </keystore>
```

4. Encrypting Keys in Configuration

This section describes an approach that enables both the symmetric and asymmetric keys on a server to be encrypted, such that traditional backup/restore procedures can be used without concern for raw key data being compromised when in transit.

The approach presented in this section is not normative. This section answers how a configuration containing secrets that are encrypted by a built-in key (Section 3) can be backup'ed from one server and restored on a different server, when each server has unique master keys. The API defined by the "ietf-keystore" YANG module presented in this document is sufficient to support the workflow described in this section.

4.1. Key Encryption Key

The ability to encrypt configured keys is predicated on the existence of a "key encryption key" (KEK). There may be any number of KEKs in a server. A KEK, by its namesake, is a key that is used to encrypt other keys. A KEK MAY be either a symmetric key or an asymmetric key.

If a KEK is a symmetric key, then the server MUST provide an API for administrators to encrypt other keys without needing to know the symmetric key's value. If the KEK is an asymmetric key, then the server SHOULD provide an API enabling the encryption of other keys or, alternatively, assume the administrators can do so themselves using the asymmetric key's public half.

A server MUST possess access to the KEK, or an API using the KEK, so that it can decrypt the other keys in the configuration at runtime.

4.2. Configuring Encrypted Keys

Each time a new key is configured, it SHOULD be encrypted by a KEK.

In "ietf-crypto-types" [I-D.ietf-netconf-crypto-types], the format for encrypted values is described by identity statements derived from the "symmetrically-encrypted-value-format" and "asymmetrically-encrypted-value-format" identity statements.

Implementations of servers implementing the "ietf-keystore" module SHOULD provide an API that simultaneously generates a key and encrypts the generated key using a KEK. Thus the cleartext value of the newly generated key may never be known to the administrators generating the keys. Such API is defined in the "ietf-ssh-common" and the "ietf-tls-common" YANG modules defined in [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server], respectively.

In case the server implementation does not provide such an API, then the generating and encrypting steps MAY be performed outside the server, e.g., by an administrator with special access control rights (e.g., an organization's crypto officer).

In either case, the encrypted key can be configured into the keystore using either the "encrypted-symmetric-key" (for symmetric keys) or the "encrypted-private-key" (for asymmetric keys) nodes. These two nodes contain both the encrypted raw key value as well as a reference to the KEK that encrypted the key.

4.3. Migrating Configuration to Another Server

When a KEK is used to encrypt other keys, migrating the configuration to another server is only possible if the second server has the same KEK. How the second server comes to have the same KEK is discussed in this section.

In some deployments, mechanisms outside the scope of this document may be used to migrate a KEK from one server to another. That said, beware that the ability to do so typically entails having access to the first server but, in some scenarios, the first server may no longer be operational.

In other deployments, an organization's crypto officer, possessing a KEK's cleartext value, configures the same KEK on the second server, presumably as a hidden key or a key protected by access-control, so that the cleartext value is not disclosed to regular administrators. However, this approach creates high-coupling to and dependency on the crypto officers that does not scale in production environments.

In order to decouple the crypto officers from the regular administrators, a special KEK, called the "master key" (MK), may be used.

A MK is commonly a globally-unique built-in (see Section 3) asymmetric key. The private raw key value, due to its long lifetime, is hidden (i.e., "hidden-private-key" in Section 2.1.4.5. of [I-D.ietf-netconf-crypto-types]). The raw public key value is often contained in an identity certificate (e.g., IDevID). How to configure a MK during the manufacturing process is outside the scope of this document.

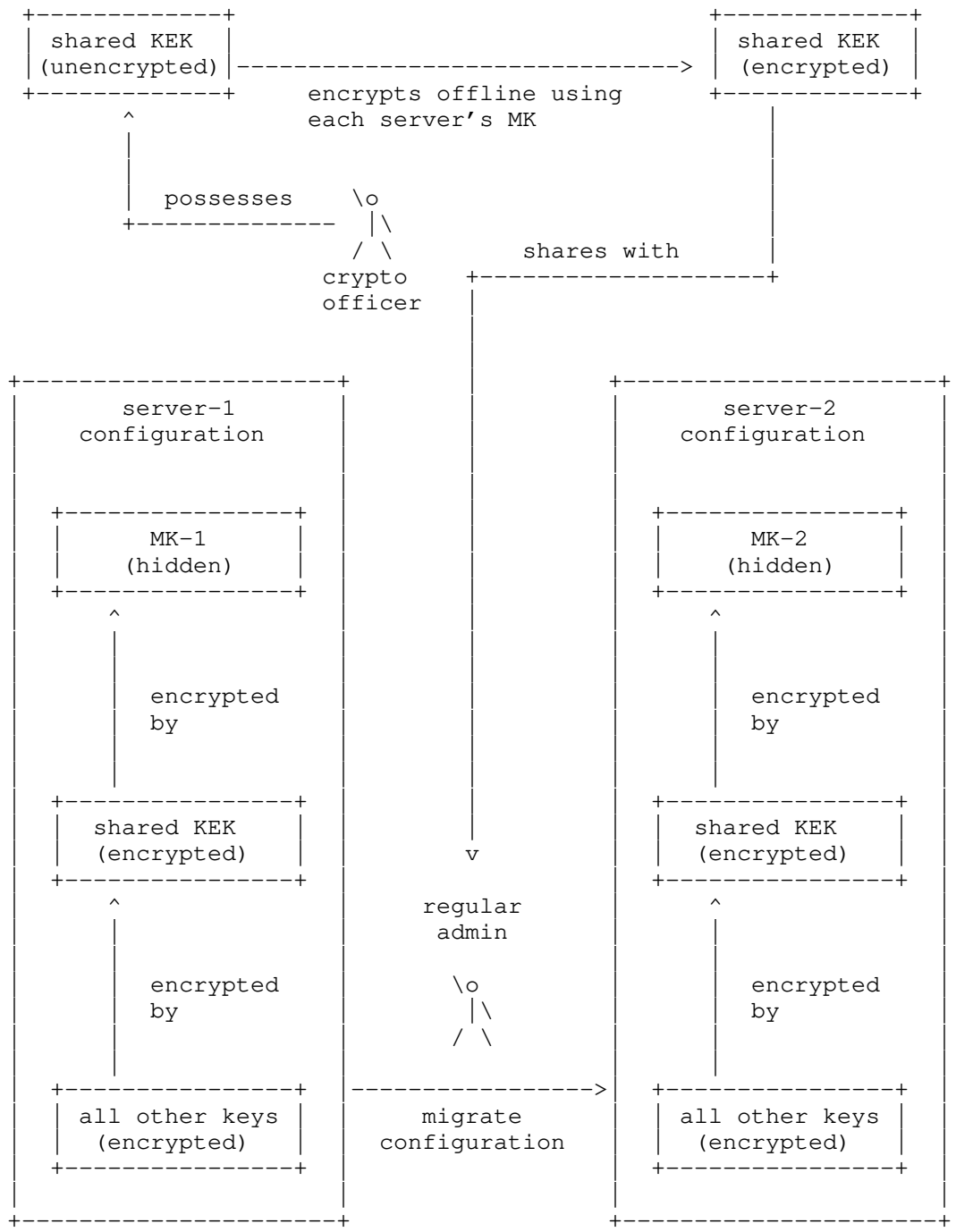
Assuming the server has a MK, the MK can be used to encrypt a "shared KEK", which is then used to encrypt the keys configured by regular administrators.

With this extra level of indirection, it is possible for a crypto officer to encrypt the same KEK for a multiplicity of servers offline using the public key contained in their identity certificates. The crypto officer can then safely handoff the encrypted KEKs to regular administrators responsible for server installations, including migrations.

In order to migrate the configuration from a first server, an administrator would need to make just a single modification to the configuration before loading it onto a second server, which is to replace the encrypted KEK keystore entry from the first server with the encrypted KEK for the second server. Upon doing this, the

configuration (containing many encrypted keys) can be loaded into the second server while enabling the second server to decrypt all the encrypted keys in the configuration.

The following diagram illustrates this idea:


shared KEK
(unencrypted)
encrypts offline using
each server's MK
shared KEK
(encrypted)
possesses
crypto
officer
shares with
server-1
configuration
MK-1
(hidden)
encrypted
by
shared KEK
(encrypted)
all other keys
(encrypted)
server-2
configuration
MK-2
(hidden)
encrypted
by
shared KEK
(encrypted)
all other keys
(encrypted)
regular
admin
migrate
configuration

5. Security Considerations

5.1. Security of Data at Rest and in Motion

The YANG module defined in this document defines a mechanism called a "keystore" that intends to protect its contents from unauthorized disclosure and modification.

In order to satisfy the expectations of a "keystore", it is RECOMMENDED that server implementations ensure that the keystore contents are encrypted when persisted to non-volatile memory, and ensure that the keystore contents that have been decrypted in volatile memory are zeroized when not in use.

The keystore contents may be encrypted either by encrypting the contents individually (e.g., using the "encrypted" value formats) or, in case cleartext values are used (which is NOT RECOMMENDED per Section 3.5 of [I-D.ietf-netconf-crypto-types]), then, e.g., disk-level encryption may be used.

If the keystore contents are not encrypted when persisted, then server implementations MUST ensure the persisted storage is inaccessible.

5.2. Unconstrained Private Key Usage

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

This module also does not constrain the usage of the associated public keys, other than in the context of a configured certificate (e.g., an identity certificate), in which case the key usage is constrained by the certificate.

5.3. Considerations for the "ietf-keystore" 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., SSH, TLS) with mutual authentication.

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

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.

Some of the readable data nodes defined in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. The following subtrees and data nodes have particular sensitivity/vulnerability:

* The "cleartext-symmetric-key" node:

  The "cleartext-symmetric-key" node, imported from the "symmetric-key-grouping" grouping defined in [I-D.ietf-netconf-crypto-types] is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" was applied to it in [I-D.ietf-netconf-crypto-types].

* The "cleartext-private-key" node:

  The "cleartext-private-key" node defined in the "asymmetric-key-pair-grouping" grouping defined in [I-D.ietf-netconf-crypto-types] is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" is applied to it in [I-D.ietf-netconf-crypto-types].

All the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "keystore" instance, may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

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

Built-in key types SHOULD be either hidden and/or encrypted (not cleartext). If this is not possible, access control mechanisms like NACM SHOULD be used to limit access to the key's secret data to only the most trusted authorized clients (e.g., belonging to an organizations crypto officer).

6. IANA Considerations

6.1. 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-keystore
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

6.2. The "YANG Module Names" Registry

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

```
name:         ietf-keystore
namespace:    urn:ietf:params:xml:ns:yang:ietf-keystore
prefix:       ks
reference:    RFC CCCC
```

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[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>.

[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>.

[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>.

[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>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-05, 21 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

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

[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>.

[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>.

[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>.

[Std-802.1AR-2018] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", August 2018, <https://standards.ieee.org/standard/802_1AR-2018.html>.

Appendix A. Change Log

A.1. 00 to 01

* Replaced the 'certificate-chain' structures with PKCS#7 structures. (Issue #1)

* Added 'private-key' as a configurable data node, and removed the 'generate-private-key' and 'load-private-key' actions. (Issue #2)

* Moved 'user-auth-credentials' to the ietf-ssh-client module. (Issues #4 and #5)

A.2. 01 to 02

* Added back 'generate-private-key' action.

* Removed 'RESTRICTED' enum from the 'private-key' leaf type.

* Fixed up a few description statements.

A.3. 02 to 03

* Changed draft's title.
* Added missing references.
* Collapsed sections and levels.
* Added RFC 8174 to Requirements Language Section.
* Renamed 'trusted-certificates' to 'pinned-certificates'.
* Changed 'public-key' from config false to config true.
* Switched 'host-key' from OneAsymmetricKey to definition from RFC 4253.

A.4. 03 to 04

* Added typedefs around leafrefs to common keystore paths
* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Removed Design Considerations section
* Moved key and certificate definitions from data tree to groupings

A.5. 04 to 05

* Removed trust anchors (now in their own draft)
* Added back global keystore structure
* Added groupings enabling keys to either be locally defined or a reference to the keystore.

A.6. 05 to 06

* Added feature "local-keys-supported"
* Added nacm:default-deny-all and nacm:default-deny-write
* Renamed generate-asymmetric-key to generate-hidden-key
* Added an install-hidden-key action
* Moved actions inside fo the "asymmetric-key" container

* Moved some groupings to draft-ietf-netconf-crypto-types

A.7. 06 to 07

* Removed a "require-instance false"

* Clarified some description statements

* Improved the keystore-usage examples

A.8. 07 to 08

* Added "inline-definition" containers to avoid posibility of the action/notification statements being under a "case" statement.

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

A.9. 08 to 09

* Added a 'description' statement to the 'must' in the /keystore/asymmetric-key node explaining that the descendant values may exist in <operational> only, and that implementation MUST assert that the values are either configured or that they exist in <operational>.

* Copied above 'must' statement (and description) into the inline-or-keystore-asymmetric-key-grouping, inline-or-keystore-asymmetric-key-with-certs-grouping, and inline-or-keystore-end-entity-cert-with-key-grouping statements.

A.10. 09 to 10

* Updated draft title to match new truststore draft title

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

* Renamed feature from 'local-keys-supported' to 'inline-definitions-supported' (same name used in truststore)

* Removed the either-all-or-none 'must' expressions for the key's 3-tuple values (since the values are now 'mandatory true' in crypto-types)

* Example updated to reflect 'mandatory true' change in crypto-types draft

A.11. 10 to 11

* Replaced typedef asymmetric-key-certificate-ref with grouping asymmetric-key-certificate-ref-grouping.

* Added feature feature 'key-generation'.

* Cloned groupings symmetric-key-grouping, asymmetric-key-pair-grouping, asymmetric-key-pair-with-cert-grouping, and asymmetric-key-pair-with-certs-grouping from crypto-keys, augmenting into each new case statements for values that have been encrypted by other keys in the keystore. Refactored keystore model to use these groupings.

* Added new 'symmetric-keys' lists, as a sibling to the existing 'asymmetric-keys' list.

* Added RPCs (not actions) 'generate-symmetric-key' and 'generate-asymmetric-key' to *return* a (potentially encrypted) key.

A.12. 11 to 12

* Updated to reflect crypto-type's draft using enumerations over identities.

* Added examples for the 'generate-symmetric-key' and 'generate-asymmetric-key' RPCs.

* Updated the Introduction section.

A.13. 12 to 13

* Updated examples to incorporate new "key-format" identities.

* Made the two "generate-*-key" RPCs be "action" statements instead.

A.14. 13 to 14

* Updated YANG module and examples to incorporate the new iana-*-algorithm modules in the crypto-types draft.

A.15. 14 to 15

* Added new "Support for Built-in Keys" section.

* Added 'must' expressions asserting that the 'key-format' leaf whenever an encrypted key is specified.

* Added inline-or-keystore-symmetric-key-grouping for PSK support.

A.16. 15 to 16

* Moved the generate key actions to ietf-crypt-types as RPCs, which are augmented by ietf-keystore to support encrypted keys. Examples updated accordingly.

* Added a SSH certificate-based key (RFC 6187) and a raw private key to the example instance document (partly so they could be referenced by examples in the SSH and TLS client/server drafts.

A.17. 16 to 17

* Removed augments to the "generate-symmetric-key" and "generate-asymmetric-key" groupings.

* Removed "generate-symmetric-key" and "generate-asymmetric-key" examples.

* Removed the "algorithm" nodes from remaining examples.

* Updated the "Support for Built-in Keys" section.

* Added new section "Encrypting Keys in Configuration".

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

A.18. 17 to 18

* Removed dangling/unnecessary ref to RFC 8342.

* r/MUST/SHOULD/ wrt strength of keys being configured over transports.

* Added an example for the "certificate-expiration" notification.

* Clarified that OS MAY have a multiplicity of underlying keystores and/or TPMs.

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

* Clarified the "Migrating Configuration to Another Server" section.

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

* Updated the Security Considerations section.

A.19. 18 to 19

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

A.20. 19 to 20

* Addressed SecDir comments from Magnus Nystroem and Sandra Murphy.

A.21. 20 to 21

* Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir.

* (Editorial) Removed the output of "grouping" statements in the tree diagrams for the "ietf-keystore" and "ex-keystore-usage" modules.

* Addressed comments raised by YANG Doctor.

A.22. 21 to 22

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

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

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

* Aligned modules with `pyang -f` formatting.

* Fixed nits found by YANG Doctor reviews.

A.23. 22 to 23

* Updated 802.1AR ref to latest version

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

* Minor editorial nits

A.24. 23 to 24

* Added features "asymmetric-keys" and "symmetric-keys"

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

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

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

A.25. 24 to 25

* Added a "term" for "key" (IEEE liaison).

* Clarified draft text to ensure proper use of the "key" term. (IEEE liaison)

* Added statement that built-in keys SHOULD NOT be cleartext. (IEEE liaison)

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

A.26. 25 to 26

* Updated per Shepherd reviews impacting the suite of drafts.

A.27. 26 to 27

* Updated per Shepherd reviews impacting the suite of drafts.

A.28. 27 to 28

* Updated per Tom Petch review.

* s/local/inline/ in feature names, grouping names, and node names.

* Removed special handling text for built-in keys

* Updated section on built-in keys to read almost the same as the section in the trust-anchors draft.

A.29. 28 to 29

* Addresses AD review comments.

* Added note to Editor to fix line foldings.

* Renamed "keystore" to "central keystore" throughout.

* Renamed "encrypted-by-choice-grouping" to "encrypted-by-grouping".

* Removed "public-key-format" and "public-key" nodes from examples.

A.30. 29 to 30

* Addresses Gen-ART review by Reese Enghardt.

* Addresses review by Tom Petch.

A.31. 30 to 31

* Addresses 1st-round of IESG reviews.

A.32. 31 to 33

* Addresses issues found in OpsDir review of the ssh-client-server draft.

* Renamed Security Considerations section s/Template for/ Considerations for/

* s/defines/presents/ in a few places.

* Add refs to where the 'operational' and 'system' datastores are defined.

A.33. 33 to 34

* Nothing changed. Only bumped for automation...

A.34. 34 to 35

* Address Roman Danyliw's comments.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Benoit Claise, Bert Wijnen, Balázs Kovács, David Lamparter, Eric Voit, Éric Vyncke, Francesca Palombini, Ladislav Lhotka, Liang Xia, Jürgen Schönwälder, Mahesh Jethanandani, Magnus Nyström, Martin Björklund, Mehmet Ersue, Murray Kucherawy, Paul Wouters, Phil Shafer, Qin Wu, Radek Krejci, Ramkumar Dhanapal, Reese Enghardt, Reshad Rahman, Rob Wilton, Roman Danyliw, Sandra Murphy, Sean Turner, Tom Petch, Warren Kumari, and Zaheduzzaman Sarker.

Author's Address

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

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 January 2021

K. Watsen
Watsen Networks

8 July 2020

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-20

Abstract

This document defines two YANG modules, one module to configure a NETCONF client and the other module to configure a NETCONF server. Both modules support both the SSH and TLS transport protocols, and support both standard NETCONF and NETCONF Call Home connections.

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 (note: not all may be present):

* "AAAA" --> the assigned RFC value for draft-ietf-netconf-crypto-types
* "BBBB" --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* "CCCC" --> the assigned RFC value for draft-ietf-netconf-keystore
* "DDDD" --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* "EEEE" --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
* "FFFF" --> the assigned RFC value for draft-ietf-netconf-tls-client-server
* "GGGG" --> the assigned RFC value for draft-ietf-netconf-http-client-server
* "HHHH" --> the assigned RFC value for this draft

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

* "2020-07-08" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

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

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

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 9 January 2021.

Copyright Notice

Table of Contents

1. Introduction

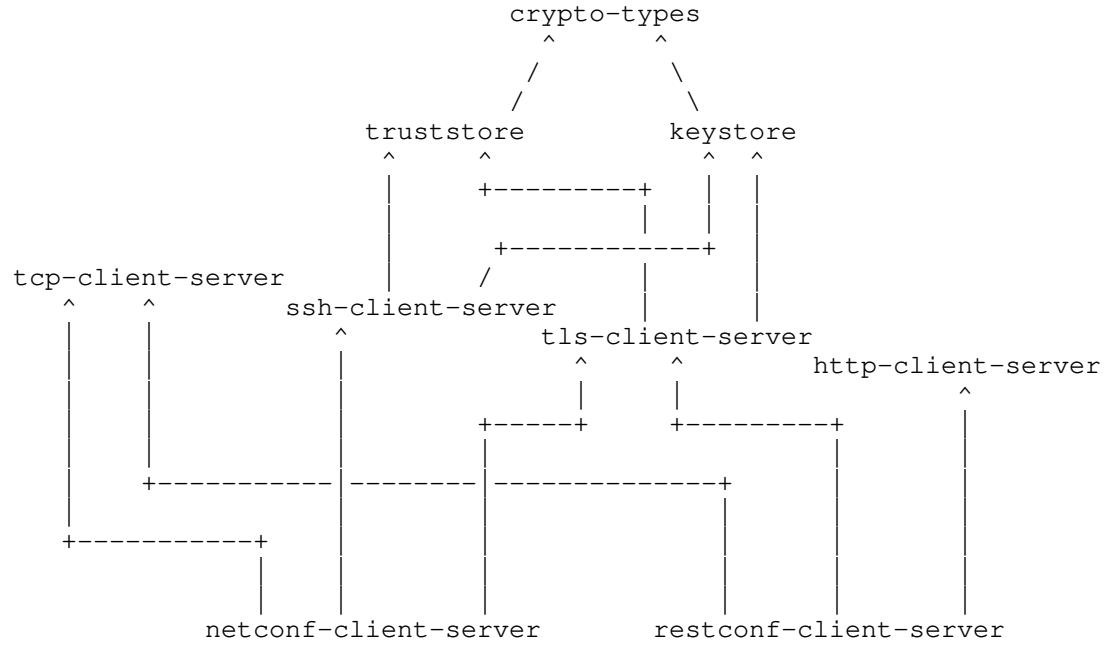
This document defines two YANG [RFC7950] modules, one module to configure a NETCONF [RFC6241] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [RFC6242] and NETCONF over TLS [RFC7589] and NETCONF Call Home connections [RFC8071].

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

2. The "ietf-netconf-client" Module

The NETCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the NETCONF client supports.

2.1. Data Model Overview

2.1.1. Features

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

```
Features:
  +-- ssh-initiate
  +-- tls-initiate
  +-- ssh-listen
  +-- tls-listen
```

2.1.2. Groupings

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

```
Groupings:
  +-- netconf-client-grouping
  +-- netconf-client-initiate-stack-grouping
  +-- netconf-client-listen-stack-grouping
  +-- netconf-client-app-grouping
```

Each of these groupings are presented in the following subsections.

2.1.2.1. The "netconf-client-grouping" Grouping

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

```
  grouping netconf-client-grouping ---> <empty>
```

Comments:

* This grouping does not define any nodes, but is maintained so that downstream modules can augment nodes into it if needed.

* The "netconf-client-grouping" defines, if it can be called that, the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

2.1.2.2. The "netconf-client-initiate-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-initiate-stack-grouping" grouping:

```
  grouping netconf-client-initiate-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-initiate}?
       |  +-- ssh
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- ssh-client-parameters
       |     |  +---u sshc:ssh-client-grouping
       |     +-- netconf-client-parameters
       |        +--u ncc:netconf-client-grouping
       +--:(tls) {tls-initiate}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- netconf-client-parameters
                +---u ncc:netconf-client-grouping
```

Comments:

* The "netconf-client-initiate-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that initiate connections to NETCONF servers, as opposed to receiving call-home [RFC8071] connections.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.3. The "netconf-client-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-listen-stack-grouping" grouping:

```
  grouping netconf-client-listen-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-listen}?
       |  +-- ssh
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- ssh-client-parameters
       |     |  +---u sshc:ssh-client-grouping
       |     +-- netconf-client-parameters
       |        +--u ncc:netconf-client-grouping
       +--:(tls) {tls-listen}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- netconf-client-parameters
                +---u ncc:netconf-client-grouping
```

Comments:

* The "netconf-client-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that receive call-home [RFC8071] connections from NETCONF servers.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.4. The "netconf-client-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-app-grouping" grouping:

```
  grouping netconf-client-app-grouping
    +-- initiate! {ssh-initiate or tls-initiate}?
    |  +-- netconf-server* [name]
    |     +-- name?                 string
    |     +-- endpoints
    |     |  +-- endpoint* [name]
    |     |     +-- name?                                        string
    |     |     +---u netconf-client-initiate-stack-grouping
    |     +-- connection-type
    |     |  +-- (connection-type)
    |     |     +--:(persistent-connection)
    |     |     |  +-- persistent!
    |     |     +--:(periodic-connection)
    |     |        +-- periodic!
    |     |           +-- period?         uint16
    |     |           +-- anchor-time?    yang:date-and-time
    |     |           +-- idle-timeout?   uint16
    |     +-- reconnect-strategy
    |        +-- start-with?     enumeration
    |        +-- max-attempts?   uint8
    +-- listen! {ssh-listen or tls-listen}?
       +-- idle-timeout?   uint16
       +-- endpoint* [name]
          +-- name?                                    string
          +---u netconf-client-listen-stack-grouping
```

Comments:

* The "netconf-client-app-grouping" defines the configuration for a NETCONF client that supports both initiating connections to NETCONF servers as well as receiving call-home connections from NETCONF servers.

* Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "netconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.
  - The "netconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

2.1.3. Protocol-accessible Nodes

The following diagram lists all the protocol-accessible nodes defined in the "ietf-netconf-client" module:

```
module: ietf-netconf-client
  +--rw netconf-client
     +---u netconf-client-app-grouping
```

Comments:

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

* For the "ietf-netconf-client" module, the protocol-accessible nodes are an instance of the "netconf-client-app-grouping" discussed in Section 2.1.2.4 grouping.

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

2.2. Example Usage

The following example illustrates configuring a NETCONF client to initiate connections, using both the SSH and TLS transport protocols, as well as to listen for call-home connections, again using both the SSH and TLS transport protocols.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

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

  <!-- NETCONF servers to initiate connections to -->
  <initiate>
    <netconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
```

```
            <ssh>
              <tcp-client-parameters>
                <remote-address>corp-fw1.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <ssh-client-parameters>
                <client-identity>
                  <username>foobar</username>
                  <public-key>
                    <keystore-reference>ssh-rsa-key</keystore-referenc\
  e>
                  </public-key>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-server-ca-certs</tru\
  ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-server-ee-certs</tru\
  ststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <max-wait>30</max-wait>
                  <max-attempts>3</max-attempts>
                </keepalives>
              </ssh-client-parameters>
              <netconf-client-parameters>
                <!-- nothing to configure -->
              </netconf-client-parameters>
            </ssh>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <tls>
              <tcp-client-parameters>
                <remote-address>corp-fw2.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
```

```
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
   ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-server-ca-certs</tru\
   ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-server-ee-certs</tru\
   ststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <netconf-client-parameters>
                <!-- nothing to configure -->
              </netconf-client-parameters>
            </tls>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
        </reconnect-strategy>
      </netconf-server>
    </initiate>

    <!-- endpoints to listen for NETCONF Call Home connections on -->
    <listen>
      <endpoint>
        <name>Intranet-facing SSH listener</name>
        <ssh>
          <tcp-server-parameters>
```

```
            <local-address>192.0.2.7</local-address>
          </tcp-server-parameters>
          <ssh-client-parameters>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <keystore-reference>ssh-rsa-key</keystore-reference>
              </public-key>
            </client-identity>
            <server-authentication>
              <ca-certs>
                <truststore-reference>trusted-server-ca-certs</truststo\
re-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-server-ee-certs</truststo\
re-reference>
              </ee-certs>
              <ssh-host-keys>
                <truststore-reference>trusted-ssh-public-keys</truststo\
re-reference>
              </ssh-host-keys>
            </server-authentication>
          </ssh-client-parameters>
          <netconf-client-parameters>
            <!-- nothing to configure -->
          </netconf-client-parameters>
        </ssh>
      </endpoint>
      <endpoint>
        <name>Intranet-facing TLS listener</name>
        <tls>
          <tcp-server-parameters>
            <local-address>192.0.2.7</local-address>
          </tcp-server-parameters>
          <tls-client-parameters>
            <client-identity>
              <certificate>
                <keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </keystore-reference>
              </certificate>
            </client-identity>
            <server-authentication>
              <ca-certs>
                <truststore-reference>trusted-server-ca-certs</truststo\
re-reference>
```

```
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-server-ee-certs</truststo\
   ore-reference>
              </ee-certs>
            </server-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-client-parameters>
          <netconf-client-parameters>
            <!-- nothing to configure -->
          </netconf-client-parameters>
        </tls>
      </endpoint>
    </listen>
  </netconf-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7589], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

<CODE BEGINS> file "ietf-netconf-client@2020-07-08.yang"

```
module ietf-netconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix ncc;

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

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
```

```
  }

  import ietf-ssh-client {
    prefix sshc;
    revision-date 2020-07-08; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    revision-date 2020-07-08; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
     Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module contains a collection of YANG definitions
     for configuring NETCONF clients.

     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 HHHH
     (https://www.rfc-editor.org/info/rfcHHHH); see the RFC
     itself for full legal notices.;

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

```
       capitals, as shown here.";

    revision 2020-07-08 {
      description
        "Initial version";
      reference
        "RFC HHHH: NETCONF Client and Server Models";
    }

    // Features

    feature ssh-initiate {
      description
        "The 'ssh-initiate' feature indicates that the NETCONF client
         supports initiating SSH connections to NETCONF servers.";
      reference
        "RFC 6242:
           Using the NETCONF Protocol over Secure Shell (SSH)";
    }

    feature tls-initiate {
      description
        "The 'tls-initiate' feature indicates that the NETCONF client
         supports initiating TLS connections to NETCONF servers.";
      reference
        "RFC 7589: Using the NETCONF Protocol over Transport
           Layer Security (TLS) with Mutual X.509 Authentication";
    }

    feature ssh-listen {
      description
        "The 'ssh-listen' feature indicates that the NETCONF client
         supports opening a port to listen for incoming NETCONF
         server call-home SSH connections.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature tls-listen {
      description
        "The 'tls-listen' feature indicates that the NETCONF client
         supports opening a port to listen for incoming NETCONF
         server call-home TLS connections.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    // Groupings
```

```
    grouping netconf-client-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         without any consideration for how underlying transport
         sessions are established.

         This grouping currently doesn't define any nodes.";
    }

    grouping netconf-client-initiate-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         'initiate' protocol stack for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-initiate";
          container ssh {
            description
              "Specifies IP and SSH specific configuration
               for the connection.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "830";
                  description
                    "The NETCONF client will attempt to connect
                     to the IANA-assigned well-known port value
                     for 'netconf-ssh' (830) if no value is
                     specified.";
                }
              }
            }
            container ssh-client-parameters {
              description
                "A wrapper around the SSH client parameters to
                 avoid name collisions.";
              uses sshc:ssh-client-grouping;
            }
            container netconf-client-parameters {
              description
                "A wrapper around the NETCONF client parameters
                 to avoid name collisions.";
```

```
            uses ncc:netconf-client-grouping;
          }
        }
      }
      case tls {
        if-feature "tls-initiate";
        container tls {
          description
            "Specifies IP and TLS specific configuration
             for the connection.";
          container tcp-client-parameters {
            description
              "A wrapper around the TCP client parameters
               to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
              refine "remote-port" {
                default "6513";
                description
                  "The NETCONF client will attempt to connect
                   to the IANA-assigned well-known port value
                   for 'netconf-tls' (6513) if no value is
                   specified.";
              }
            }
          }
          container tls-client-parameters {
            must "client-identity" {
              description
                "NETCONF/TLS clients MUST pass some
                 authentication credentials.";
            }
            description
              "A wrapper around the TLS client parameters
               to avoid name collisions.";
            uses tlsc:tls-client-grouping;
          }
          container netconf-client-parameters {
            description
              "A wrapper around the NETCONF client parameters
               to avoid name collisions.";
            uses ncc:netconf-client-grouping;
          }
        }
      }
    }
  } // netconf-client-initiate-stack-grouping

  grouping netconf-client-listen-stack-grouping {
```

```
      description
        "A reusable grouping for configuring a NETCONF client
         'listen' protocol stack for a single connection.  The
         'listen' stack supports call home connections, as
         described in RFC 8071";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-listen";
          container ssh {
            description
              "SSH-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4334";
                  description
                    "The NETCONF client will listen on the IANA-
                     assigned well-known port for 'netconf-ch-ssh'
                     (4334) if no value is specified.";
                }
              }
            }
            container ssh-client-parameters {
              description
                "A wrapper around the SSH client parameters
                 to avoid name collisions.";
              uses sshc:ssh-client-grouping;
            }
            container netconf-client-parameters {
              description
                "A wrapper around the NETCONF client parameters
                 to avoid name collisions.";
              uses ncc:netconf-client-grouping;
            }
          }
        }
        case tls {
          if-feature "tls-listen";
          container tls {
```

```
            description
              "TLS-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4334";
                  description
                    "The NETCONF client will listen on the IANA-
                      assigned well-known port for 'netconf-ch-ssh'
                      (4334) if no value is specified.";
                }
              }
            }
            container tls-client-parameters {
              must "client-identity" {
                description
                  "NETCONF/TLS clients MUST pass some
                   authentication credentials.";
              }
              description
                "A wrapper around the TLS client parameters
                 to avoid name collisions.";
              uses tlsc:tls-client-grouping;
            }
            container netconf-client-parameters {
              description
                "A wrapper around the NETCONF client parameters
                 to avoid name collisions.";
              uses ncc:netconf-client-grouping;
            }
          }
        }
      }
    } // netconf-client-listen-stack-grouping

    grouping netconf-client-app-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         application that supports both 'initiate' and 'listen'
         protocol stacks for a multiplicity of connections.";
      container initiate {
        if-feature "ssh-initiate or tls-initiate";
        presence "Enables client to initiate TCP connections";
        description
```

```
          "Configures client initiating underlying TCP connections.";
        list netconf-server {
          key "name";
          min-elements 1;
          description
            "List of NETCONF servers the NETCONF client is to
             maintain simultaneous connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the NETCONF server.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
              description
                "A user-ordered list of endpoints that the NETCONF
                 client will attempt to connect to in the specified
                 sequence.  Defining more than one enables
                 high-availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for the endpoint.";
              }
              uses netconf-client-initiate-stack-grouping;
            } // list endpoint
          } // container endpoints

          container connection-type {
            description
              "Indicates the NETCONF client's preference for how the
               NETCONF connection is maintained.";
            choice connection-type {
              mandatory true;
              description
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
                  presence "Indicates that a persistent connection is
                            to be maintained.";
                  description
                    "Maintain a persistent connection to the NETCONF
                     server.  If the connection goes down, immediately
```

```
                    start trying to reconnect to the NETCONF server,
                    using the reconnection strategy.

                    This connection type minimizes any NETCONF server
                    to NETCONF client data-transfer delay, albeit at
                    the expense of holding resources longer.";
                }
              }
              case periodic-connection {
                container periodic {
                  presence "Indicates that a periodic connection is
                            to be maintained.";
                  description
                    "Periodically connect to the NETCONF server.

                     This connection type increases resource
                     utilization, albeit with increased delay in
                     NETCONF server to NETCONF client interactions.

                     The NETCONF client should close the underlying
                     TCP connection upon completing planned activities.

                     In the case that the previous connection is still
                     active, establishing a new connection is NOT
                     RECOMMENDED.";
                  leaf period {
                    type uint16;
                    units "minutes";
                    default "60";
                    description
                      "Duration of time between periodic connections.";
                  }
                  leaf anchor-time {
                    type yang:date-and-time {
                      // constrained to minute-level granularity
                      pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                            + '(Z|[\+\-]\d{2}:\d{2})';
                    }
                    description
                      "Designates a timestamp before or after which a
                       series of periodic connections are determined.
                       The periodic connections occur at a whole
                       multiple interval from the anchor time.  For
                       example, for an anchor time is 15 minutes past
                       midnight and a period interval of 24 hours, then
                       a periodic connection will occur 15 minutes past
                       midnight everyday.";
                  }
```

```
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default 120; // two minutes
                  description
                    "Specifies the maximum number of seconds that
                     a NETCONF session may remain idle. A NETCONF
                     session will be dropped if it is idle for an
                     interval longer then this number of seconds.
                     If set to zero, then the NETCONF client will
                     never drop a session because it is idle.";
                }
              }
            }
          }
        }
        container reconnect-strategy {
          description
            "The reconnection strategy directs how a NETCONF client
             reconnects to a NETCONF server, after discovering its
             connection to the server has dropped, even if due to a
             reboot.  The NETCONF client starts with the specified
             endpoint and tries to connect to it max-attempts times
             before trying the next endpoint in the list (round
             robin).";
          leaf start-with {
            type enumeration {
              enum first-listed {
                description
                  "Indicates that reconnections should start with
                   the first endpoint listed.";
              }
              enum last-connected {
                description
                  "Indicates that reconnections should start with
                   the endpoint last connected to.  If no previous
                   connection has ever been established, then the
                   first endpoint configured is used.   NETCONF
                   clients SHOULD be able to remember the last
                   endpoint connected to across reboots.";
              }
              enum random-selection {
                description
                  "Indicates that reconnections should start with
                   a random endpoint.";
              }
            }
            default "first-listed";
```

```
            description
              "Specifies which of the NETCONF server's endpoints
               the NETCONF client should start with when trying
               to connect to the NETCONF server.";
          }
          leaf max-attempts {
            type uint8 {
              range "1..max";
            }
            default "3";
            description
              "Specifies the number times the NETCONF client tries
               to connect to a specific endpoint before moving on
               to the next endpoint in the list (round robin).";
          }
        }
      } // netconf-server
    } // initiate

    container listen {
      if-feature "ssh-listen or tls-listen";
      presence "Enables client to accept call-home connections";
      description
        "Configures the client to accept call-home TCP connections.";
      leaf idle-timeout {
        type uint16;
        units "seconds";
        default "3600"; // one hour
        description
          "Specifies the maximum number of seconds that a NETCONF
           session may remain idle. A NETCONF session will be
           dropped if it is idle for an interval longer than this
           number of seconds.  If set to zero, then the server
           will never drop a session because it is idle.  Sessions
           that have a notification subscription active are never
           dropped.";
      }
      list endpoint {
        key "name";
        min-elements 1;
        description
          "List of endpoints to listen for NETCONF connections.";
        leaf name {
          type string;
          description
            "An arbitrary name for the NETCONF listen endpoint.";
        }
        uses netconf-client-listen-stack-grouping;
```

```
        } // endpoint
      } // listen
    } // netconf-client-app-grouping

    // Protocol accessible node, for servers that implement
    // this module.
    container netconf-client {
      uses netconf-client-app-grouping;
      description
        "Top-level container for NETCONF client configuration.";
    }
  }

  <CODE ENDS>
```

3. The "ietf-netconf-server" Module

The NETCONF server model presented in this section supports both listening for connections as well as initiating call-home connections, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the NETCONF server supports.

3.1. Data Model Overview

3.1.1. Features

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

```
Features:
  +-- ssh-listen
  +-- tls-listen
  +-- ssh-call-home
  +-- tls-call-home
```

3.1.2. Groupings

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

```
Groupings:
  +-- netconf-server-grouping
  +-- netconf-server-listen-stack-grouping
  +-- netconf-server-callhome-stack-grouping
  +-- netconf-server-app-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "netconf-server-grouping" Grouping

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

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

  grouping netconf-server-grouping
    +-- client-identity-mappings
            {(tls-listen or tls-call-home) and (sshcmn:ssh-x509-cert\
s)}?
       +---u x509c2n:cert-to-name
```

Comments:

* The "netconf-server-grouping" defines the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).

* The "client-identity-mappings" node, which must be enabled by "feature" statements, defines a mapping from certificate fields to NETCONF user names.

* For the referenced grouping statement(s):

  - The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

3.1.2.2. The "netconf-server-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-listen-stack-grouping" grouping:

```
  grouping netconf-server-listen-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-listen}?
       |  +-- ssh
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- ssh-server-parameters
       |     |  +---u sshs:ssh-server-grouping
       |     +-- netconf-server-parameters
       |        +---u ncs:netconf-server-grouping
       +--:(tls) {tls-listen}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- netconf-server-parameters
                +---u ncs:netconf-server-grouping
```

Comments:

* The "netconf-server-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack for NETCONF servers that listen for standard connections from NETCONF clients, as opposed to initiating call-home [RFC8071] connections.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.3. The "netconf-server-callhome-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-callhome-stack-grouping" grouping:

```
  grouping netconf-server-callhome-stack-grouping
    +-- (transport)
       +--:(ssh) {ssh-call-home}?
       |  +-- ssh
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- ssh-server-parameters
       |     |  +---u sshs:ssh-server-grouping
       |     +-- netconf-server-parameters
       |        +---u ncs:netconf-server-grouping
       +--:(tls) {tls-call-home}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- netconf-server-parameters
                +---u ncs:netconf-server-grouping
```

Comments:

* The "netconf-server-callhome-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF servers that initiate call-home [RFC8071] connections to NETCONF clients.

* The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.4. The "netconf-server-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-app-grouping" grouping:

```
  grouping netconf-server-app-grouping
    +-- listen! {ssh-listen or tls-listen}?
    |  +-- idle-timeout?   uint16
    |  +-- endpoint* [name]
    |     +-- name?                                    string
    |     +---u netconf-server-listen-stack-grouping
    +-- call-home! {ssh-call-home or tls-call-home}?
       +-- netconf-client* [name]
          +-- name?                 string
          +-- endpoints
          |  +-- endpoint* [name]
          |     +-- name?                                        string
          |     +---u netconf-server-callhome-stack-grouping
          +-- connection-type
          |  +-- (connection-type)
          |     +--:(persistent-connection)
          |     |  +-- persistent!
          |     +--:(periodic-connection)
          |        +-- periodic!
          |           +-- period?         uint16
          |           +-- anchor-time?    yang:date-and-time
          |           +-- idle-timeout?   uint16
          +-- reconnect-strategy
             +-- start-with?     enumeration
             +-- max-attempts?   uint8
```

Comments:

* The "netconf-server-app-grouping" defines the configuration for a NETCONF server that supports both listening for connections from NETCONF clients as well as initiatiating call-home connections to NETCONF clients.

* Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "netconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
  - The "netconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following diagram lists all the protocol-accessible nodes defined in the "ietf-netconf-server" module:

```
module: ietf-netconf-server
  +--rw netconf-server
     +---u netconf-server-app-grouping
```

Comments:

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

* For the "ietf-netconf-server" module, the protocol-accessible nodes are an instance of the "netconf-server-app-grouping" discussed in Section 3.1.2.4 grouping.

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

3.2. Example Usage

The following example illustrates configuring a NETCONF server to listen for NETCONF client connections using both the SSH and TLS transport protocols, as well as configuring call-home to two NETCONF clients, one using SSH and the other using TLS.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

<netconf-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">

  <!-- endpoints to listen for NETCONF connections on -->
  <listen>
    <endpoint> <!-- listening for SSH connections -->
      <name>netconf/ssh</name>
      <ssh>
        <tcp-server-parameters>
          <local-address>192.0.2.7</local-address>
        </tcp-server-parameters>
        <ssh-server-parameters>
          <server-identity>
            <host-key>
```

```
                <name>deployment-specific-certificate</name>
                <public-key>
                  <keystore-reference>ssh-rsa-key</keystore-reference>
                </public-key>
              </host-key>
            </server-identity>
            <client-authentication>
              <supported-authentication-methods>
                <publickey/>
              </supported-authentication-methods>
            </client-authentication>
          </ssh-server-parameters>
          <netconf-server-parameters>
            <!-- nothing to configure -->
          </netconf-server-parameters>
        </ssh>
      </endpoint>
      <endpoint> <!-- listening for TLS sessions -->
        <name>netconf/tls</name>
        <tls>
          <tcp-server-parameters>
            <local-address>192.0.2.7</local-address>
          </tcp-server-parameters>
          <tls-server-parameters>
            <server-identity>
              <certificate>
                <keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </keystore-reference>
              </certificate>
            </server-identity>
            <client-authentication>
              <ca-certs>
                <truststore-reference>trusted-client-ca-certs</truststore-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-client-ee-certs</truststore-reference>
              </ee-certs>
            </client-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-server-parameters>
          <netconf-server-parameters>
            <client-identity-mappings>
```

```
              <cert-to-name>
                <id>1</id>
                <fingerprint>11:0A:05:11:00</fingerprint>
                <map-type>x509c2n:specified</map-type>
                <name>scooby-doo</name>
              </cert-to-name>
              <cert-to-name>
                <id>2</id>
                <map-type>x509c2n:san-any</map-type>
              </cert-to-name>
            </client-identity-mappings>
          </netconf-server-parameters>
        </tls>
      </endpoint>
    </listen>

    <!-- calling home to SSH and TLS based NETCONF clients -->
    <call-home>
      <netconf-client> <!-- SSH-based client -->
        <name>config-mgr</name>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <ssh>
              <tcp-client-parameters>
                <remote-address>east.config-mgr.example.com</remote-ad\
dress>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <ssh-server-parameters>
                <server-identity>
                  <host-key>
                    <name>deployment-specific-certificate</name>
                    <public-key>
                      <keystore-reference>ssh-rsa-key</keystore-refere\
nce>
                    </public-key>
                  </host-key>
                </server-identity>
                <client-authentication>
                  <supported-authentication-methods>
                    <publickey/>
                  </supported-authentication-methods>
                </client-authentication>
```

```
              </ssh-server-parameters>
              <netconf-server-parameters>
                <!-- nothing to configure -->
              </netconf-server-parameters>
            </ssh>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <ssh>
              <tcp-client-parameters>
                <remote-address>west.config-mgr.example.com</remote-ad\
   dress>
              </tcp-client-parameters>
              <ssh-server-parameters>
                <server-identity>
                  <host-key>
                    <name>deployment-specific-certificate</name>
                    <public-key>
                      <keystore-reference>ssh-rsa-key</keystore-refere\
   nce>
                    </public-key>
                  </host-key>
                </server-identity>
                <client-authentication>
                  <supported-authentication-methods>
                    <publickey/>
                  </supported-authentication-methods>
                </client-authentication>
              </ssh-server-parameters>
              <netconf-server-parameters>
                <!-- nothing to configure -->
              </netconf-server-parameters>
            </ssh>
          </endpoint>
        </endpoints>
        <connection-type>
          <periodic>
            <idle-timeout>300</idle-timeout>
            <period>60</period>
          </periodic>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </netconf-client>
      <netconf-client> <!-- TLS-based client -->
        <name>data-collector</name>
```

```
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <tls>
              <tcp-client-parameters>
                <remote-address>east.analytics.example.com</remote-add\
ress>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <netconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
```

```
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </netconf-server-parameters>
            </tls>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <tls>
              <tcp-client-parameters>
                <remote-address>west.analytics.example.com</remote-add\
ress>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <netconf-server-parameters>
```

```
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </netconf-server-parameters>
            </tls>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
        <reconnect-strategy>
          <start-with>first-listed</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </netconf-client>
    </call-home>
  </netconf-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7407], [RFC7589], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

<CODE BEGINS> file "ietf-netconf-server@2020-07-08.yang"

```
module ietf-netconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix ncs;

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

```
  import ietf-x509-cert-to-name {
    prefix x509c2n;
    reference
      "RFC 7407: A YANG Data Model for SNMP Configuration";
  }

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2020-07-08; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-ssh-server {
    prefix sshs;
    revision-date 2020-07-08; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-server {
    prefix tlss;
    revision-date 2020-07-08; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
     Author:   Gary Wu <mailto:garywu@cisco.com>
     Author:   Juergen Schoenwaelder
```

```
                <mailto:j.schoenwaelder@jacobs-university.de>";

    description
      "This module contains a collection of YANG definitions
       for configuring NETCONF servers.

       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 HHHH
       (https://www.rfc-editor.org/info/rfcHHHH); see the RFC
       itself for full legal notices.;

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

    revision 2020-07-08 {
      description
        "Initial version";
      reference
        "RFC HHHH: NETCONF Client and Server Models";
    }

    // Features

    feature ssh-listen {
      description
        "The 'ssh-listen' feature indicates that the NETCONF server
         supports opening a port to accept NETCONF over SSH
         client connections.";
      reference
        "RFC 6242:
           Using the NETCONF Protocol over Secure Shell (SSH)";
    }

    feature tls-listen {
      description
```

```
        "The 'tls-listen' feature indicates that the NETCONF server
         supports opening a port to accept NETCONF over TLS
         client connections.";
      reference
        "RFC 7589: Using the NETCONF Protocol over Transport
                   Layer Security (TLS) with Mutual X.509
                   Authentication";
    }

    feature ssh-call-home {
      description
        "The 'ssh-call-home' feature indicates that the NETCONF
         server supports initiating a NETCONF over SSH call
         home connection to NETCONF clients.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature tls-call-home {
      description
        "The 'tls-call-home' feature indicates that the NETCONF
         server supports initiating a NETCONF over TLS call
         home connection to NETCONF clients.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    // Groupings

    grouping netconf-server-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         without any consideration for how underlying transport
         sessions are established.

         Note that this grouping uses a fairly typical descendent
         node name such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue by wrapping the 'uses'
         statement in a container called, e.g.,
         'netconf-server-parameters'.  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container client-identity-mappings {
        if-feature
          "(tls-listen or tls-call-home) and (sshcmn:ssh-x509-certs)";
        description
```

```
          "Specifies mappings through which NETCONF client X.509
           certificates are used to determine a NETCONF username.
           If no matching and valid cert-to-name list entry can be
           found, then the NETCONF server MUST close the connection,
           and MUST NOT accept NETCONF messages over it.";
        reference
          "RFC 7407: A YANG Data Model for SNMP Configuration.";
        uses x509c2n:cert-to-name {
          refine "cert-to-name/fingerprint" {
            mandatory false;
            description
              "A 'fingerprint' value does not need to be specified
               when the 'cert-to-name' mapping is independent of
               fingerprint matching.  A 'cert-to-name' having no
               fingerprint value will match any client certificate
               and therefore should only be present at the end of
               the user-ordered 'cert-to-name' list.";
          }
        }
      }
    }

    grouping netconf-server-listen-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         'listen' protocol stack for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-listen";
          container ssh {
            description
              "SSH-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "830";
                  description
                    "The NETCONF server will listen on the
                     IANA-assigned well-known port value
                     for 'netconf-ssh' (830) if no value
                     is specified.";
```

```
              }
            }
          }
          container ssh-server-parameters {
            description
              "A wrapper around the SSH server parameters
               to avoid name collisions.";
            uses sshs:ssh-server-grouping;
          }
          container netconf-server-parameters {
            description
              "A wrapper around the NETCONF server parameters
               to avoid name collisions.";
            uses ncs:netconf-server-grouping;
          }
        }
      }
      case tls {
        if-feature "tls-listen";
        container tls {
          description
            "TLS-specific listening configuration for inbound
             connections.";
          container tcp-server-parameters {
            description
              "A wrapper around the TCP client parameters
               to avoid name collisions.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "6513";
                description
                  "The NETCONF server will listen on the
                   IANA-assigned well-known port value
                   for 'netconf-tls' (6513) if no value
                   is specified.";
              }
            }
          }
          container tls-server-parameters {
            description
              "A wrapper around the TLS server parameters to
               avoid name collisions.";
            uses tlss:tls-server-grouping {
              refine "client-authentication" {
              must 'ca-certs or ee-certs';
                description
                  "NETCONF/TLS servers MUST validate client
                   certificates.  This configures certificates
```

```
                    at the socket-level (i.e. bags), more
                    discriminating client-certificate checks
                    SHOULD be implemented by the application.";
                  reference
                    "RFC 7589:
                     Using the NETCONF Protocol over Transport Layer
                     Security (TLS) with Mutual X.509 Authentication";
                }
              }
            }
            container netconf-server-parameters {
              description
                "A wrapper around the NETCONF server parameters
                 to avoid name collisions.";
              uses ncs:netconf-server-grouping;
            }
          }
        }
      }
    }

    grouping netconf-server-callhome-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         'call-home' protocol stack, for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-call-home";
          container ssh {
            description
              "Specifies SSH-specific call-home transport
               configuration.";
            container tcp-client-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "4334";
                  description
                    "The NETCONF server will attempt to connect
                     to the IANA-assigned well-known port for
                     'netconf-ch-tls' (4334) if no value is
                     specified.";
                }
```

```
            }
          }
          container ssh-server-parameters {
            description
              "A wrapper around the SSH server parameters
               to avoid name collisions.";
            uses sshs:ssh-server-grouping;
          }
          container netconf-server-parameters {
            description
              "A wrapper around the NETCONF server parameters
               to avoid name collisions.";
            uses ncs:netconf-server-grouping;
          }
        }
      }
      case tls {
        if-feature "tls-call-home";
        container tls {
          description
            "Specifies TLS-specific call-home transport
             configuration.";
          container tcp-client-parameters {
            description
              "A wrapper around the TCP client parameters
               to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
              refine "remote-port" {
                default "4335";
                description
                  "The NETCONF server will attempt to connect
                   to the IANA-assigned well-known port for
                   'netconf-ch-tls' (4335) if no value is
                   specified.";
              }
            }
          }
          container tls-server-parameters {
            description
              "A wrapper around the TLS server parameters to
               avoid name collisions.";
            uses tlss:tls-server-grouping {
              refine "client-authentication" {
              must 'ca-certs or ee-certs';
                description
                  "NETCONF/TLS servers MUST validate client
                   certificates.  This configures certificates
                   at the socket-level (i.e. bags), more
```

```
                    discriminating client-certificate checks
                    SHOULD be implemented by the application.";
                  reference
                    "RFC 7589:
                     Using the NETCONF Protocol over Transport Layer
                     Security (TLS) with Mutual X.509 Authentication";
                }
              }
            }
            container netconf-server-parameters {
              description
                "A wrapper around the NETCONF server parameters
                 to avoid name collisions.";
              uses ncs:netconf-server-grouping;
            }
          }
        }
      }
    }

    grouping netconf-server-app-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         application that supports both 'listen' and 'call-home'
         protocol stacks for a multiplicity of connections.";
      container listen {
        if-feature "ssh-listen or tls-listen";
        presence
          "Enables server to listen for NETCONF client connections.";
        description
          "Configures listen behavior";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default 3600; // one hour
          description
            "Specifies the maximum number of seconds that a NETCONF
             session may remain idle. A NETCONF session will be
             dropped if it is idle for an interval longer than this
             number of seconds.  If set to zero, then the server
             will never drop a session because it is idle.  Sessions
             that have a notification subscription active are never
             dropped.";
        }
        list endpoint {
          key "name";
          min-elements 1;
          description
```

```
            "List of endpoints to listen for NETCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the NETCONF listen endpoint.";
          }
          uses netconf-server-listen-stack-grouping;
        }
      }
      container call-home {
        if-feature "ssh-call-home or tls-call-home";
        presence
          "Enables the NETCONF server to initiate the underlying
           transport connection to NETCONF clients.";
        description "Configures call home behavior.";
        list netconf-client {
          key "name";
          min-elements 1;
          description
            "List of NETCONF clients the NETCONF server is to
             maintain simultaneous call-home connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the remote NETCONF client.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
              description
                "A non-empty user-ordered list of endpoints for this
                 NETCONF server to try to connect to in sequence.
                 Defining more than one enables high-availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this endpoint.";
              }
              uses netconf-server-callhome-stack-grouping;
            }
          }
          container connection-type {
            description
              "Indicates the NETCONF server's preference for how the
```

```
           NETCONF connection is maintained.";
         choice connection-type {
           mandatory true;
           description
             "Selects between available connection types.";
           case persistent-connection {
             container persistent {
               presence "Indicates that a persistent connection is
                         to be maintained.";
               description
                 "Maintain a persistent connection to the NETCONF
                  client. If the connection goes down, immediately
                  start trying to reconnect to the NETCONF client,
                  using the reconnection strategy.

                  This connection type minimizes any NETCONF client
                  to NETCONF server data-transfer delay, albeit at
                  the expense of holding resources longer.";
             }
           }
           case periodic-connection {
             container periodic {
               presence "Indicates that a periodic connection is
                         to be maintained.";
               description
                 "Periodically connect to the NETCONF client.

                  This connection type increases resource
                  utilization, albeit with increased delay in
                  NETCONF client to NETCONF client interactions.

                  The NETCONF client SHOULD gracefully close the
                  connection using <close-session> upon completing
                  planned activities.  If the NETCONF session is
                  not closed gracefully, the NETCONF server MUST
                  immediately attempt to reestablish the connection.

                  In the case that the previous connection is still
                  active (i.e., the NETCONF client has not closed
                  it yet), establishing a new connection is NOT
                  RECOMMENDED.";
               leaf period {
                 type uint16;
                 units "minutes";
                 default "60";
                 description
                   "Duration of time between periodic connections.";
               }
```

```
                leaf anchor-time {
                  type yang:date-and-time {
                    // constrained to minute-level granularity
                    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                          + '(Z|[\+\-]\d{2}:\d{2})';
                  }
                  description
                    "Designates a timestamp before or after which a
                     series of periodic connections are determined.
                     The periodic connections occur at a whole
                     multiple interval from the anchor time.  For
                     example, for an anchor time is 15 minutes past
                     midnight and a period interval of 24 hours, then
                     a periodic connection will occur 15 minutes past
                     midnight everyday.";
                }
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default 120; // two minutes
                  description
                    "Specifies the maximum number of seconds that
                     a NETCONF session may remain idle. A NETCONF
                     session will be dropped if it is idle for an
                     interval longer than this number of seconds.
                     If set to zero, then the server will never
                     drop a session because it is idle.";
                }
              }
            } // case periodic-connection
          } // choice connection-type
        } // container connection-type
        container reconnect-strategy {
          description
            "The reconnection strategy directs how a NETCONF server
             reconnects to a NETCONF client, after discovering its
             connection to the client has dropped, even if due to a
             reboot.  The NETCONF server starts with the specified
             endpoint and tries to connect to it max-attempts times
             before trying the next endpoint in the list (round
             robin).";
          leaf start-with {
            type enumeration {
              enum first-listed {
                description
                  "Indicates that reconnections should start with
                   the first endpoint listed.";
              }
```

```
                enum last-connected {
                  description
                    "Indicates that reconnections should start with
                     the endpoint last connected to.  If no previous
                     connection has ever been established, then the
                     first endpoint configured is used.   NETCONF
                     servers SHOULD be able to remember the last
                     endpoint connected to across reboots.";
                }
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default "first-listed";
              description
                "Specifies which of the NETCONF client's endpoints
                 the NETCONF server should start with when trying
                 to connect to the NETCONF client.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the NETCONF server tries
                 to connect to a specific endpoint before moving on
                 to the next endpoint in the list (round robin).";
            }
          } // container reconnect-strategy
        } // list netconf-client
      } // container call-home
    } // grouping netconf-server-app-grouping

    // Protocol accessible node, for servers that implement
    // this module.
    container netconf-server {
      uses netconf-server-app-grouping;
      description
        "Top-level container for NETCONF server configuration.";
    }
  }

  <CODE ENDS>
```

4. Security Considerations

4.1. The "ietf-netconf-client" YANG Module

The "ietf-netconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

4.2. The "ietf-netconf-server" YANG Module

The "ietf-netconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

5. IANA Considerations

5.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-netconf-client
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-server
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 two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registrations are requested:

```
name:         ietf-netconf-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-client
prefix:       ncc
reference:    RFC HHHH

name:         ietf-netconf-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-server
prefix:       ncs
reference:    RFC HHHH
```

6. References

6.1. Normative References

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

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

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

[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>.

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[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>.

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

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

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

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

[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>.

Appendix A. Change Log

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

A.1. 00 to 01

* Renamed "keychain" to "keystore".

A.2. 01 to 02

* Added to ietf-netconf-client ability to connected to a cluster of endpoints, including a reconnection-strategy.

* Added to ietf-netconf-client the ability to configure connection-type and also keep-alive strategy.

* Updated both modules to accommodate new groupings in the ssh/tls drafts.

A.3. 02 to 03

* Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

* Changed 'netconf-client' to be a grouping (not a container).

A.4. 03 to 04

* Added RFC 8174 to Requirements Language Section.

* Replaced refine statement in ietf-netconf-client to add a mandatory true.

* Added refine statement in ietf-netconf-server to add a must statement.

* Now there are containers and groupings, for both the client and server models.

A.5. 04 to 05

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated examples to inline key and certificates (no longer a leafref to keystore)

A.6. 05 to 06

* Fixed change log missing section issue.

* Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.

* Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

* Removed "idle-timeout" from "persistent" connection config.

* Added "random-selection" for reconnection-strategy's "starts-with" enum.

* Replaced "connection-type" choice default (persistent) with "mandatory true".

* Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.

* Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

* Modified examples to be compatible with new crypto-types algs

A.9. 08 to 09

* Corrected use of "mandatory true" for "address" leafs.

* Updated examples to reflect update to groupings defined in the keystore draft.

* Updated to use groupings defined in new TCP and HTTP drafts.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

* Reformatted YANG modules.

A.11. 10 to 11

* Adjusted for the top-level "demux container" added to groupings imported from other modules.

* Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Moved "expanded" tree diagrams to the Appendix.

A.12. 11 to 12

* Removed the "Design Considerations" section.

* Removed the 'must' statement limiting keepalives in periodic connections.

* Updated models and examples to reflect removal of the "demux" containers in the imported models.

* Updated the "periodic-connnection" description statements to be more like the RESTCONF draft, especially where it described dropping the underlying TCP connection.

* Updated text to better reference where certain examples come from (e.g., which Section in which draft).

* In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.

* Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

A.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

A.14. 13 to 14

* Adjusting from change in TLS client model (removing the top-level 'certificate' container), by swapping refining-in a 'mandatory true' statement with a 'must' statement outside the 'uses' statement.

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

A.15. 14 to 15

* Refactored both the client and server modules similar to how the ietf-restconf-server module was refactored in -13 of that draft, and the ietf-restconf-client grouping.

A.16. 15 to 16

* Added refinement to make "cert-to-name/fingerprint" be mandatory false.

* Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

A.17. 16 to 17

* Updated examples to include the "*-key-format" nodes.

* Updated examples to remove the "required" nodes.

* Updated examples to remove the "client-auth-defined-elsewhere" nodes.

A.18. 17 to 18

* Updated examples to reflect new "bag" addition to truststore.

A.19. 18 to 19

* Updated examples to remove the 'algorithm' nodes.

* Updated examples to reflect the new TLS keepalives structure.

* Added keepalives to the tcp-client-parameters section in the netconf-server SSH-based call-home example.

* Added a TLS-based call-home example to the netconf-client example.

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

A.20. 19 to 20

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

* Removed expanded tree diagrams that were listed in the Appendix.

* Updated the Security Considerations section.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Ramkumar Dhanapal, Mehmet Ersue, Balazs Kovacs, David Lamparter, Ladislav Lhotka, Alan Luchuk, Radek Krejci, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, and Bert Wijnen.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-36

Abstract

This document presents two YANG modules, one module to configure a NETCONF client and the other module to configure a NETCONF server. Both modules support both the SSH and TLS transport protocols, and support both standard NETCONF and NETCONF Call Home connections.

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 (note: not all may be present):

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* EEEE --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
* GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server
* HHHH --> 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:

* 2024-03-16 --> the publication date of this draft

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

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A. Change Log

Status of This Memo

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

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

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

This Internet-Draft will expire on 17 September 2024.

Table of Contents

1. Introduction

This document presents two YANG [RFC7950] modules, one module to configure a NETCONF [RFC6241] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [RFC6242] and NETCONF over TLS [RFC7589] and NETCONF Call Home connections [RFC8071].

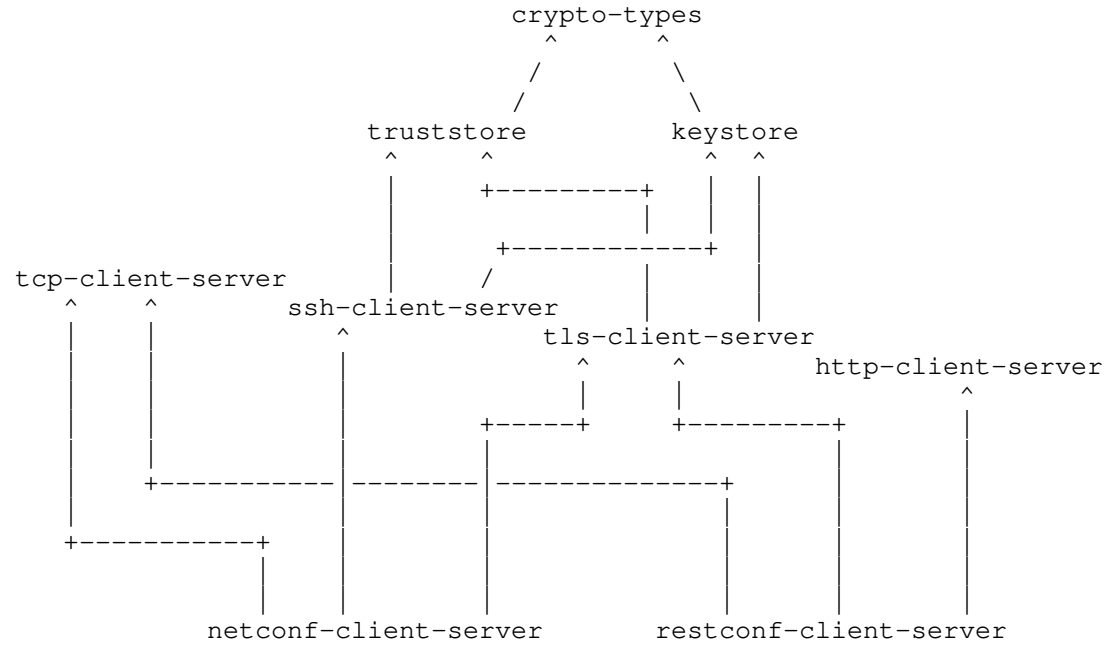
1.1. Relation to other RFCs

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

The dependency relationship between the primary YANG groupings defined in the various RFCs is presented in the below diagram. In some cases, a draft may define secondary groupings that introduce

dependencies not illustrated in the diagram. The labels in the diagram are a shorthand name for the defining RFC. The citation reference for shorthand name is 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.



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

Table 1: Label 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, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational> (Section 5.3 of [RFC8342]), and <system> [I-D.ietf-netmod-system-config], if implemented.

1.4. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-netconf-client" Module

The NETCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the NETCONF client supports.

2.1. Data Model Overview

This section provides an overview of the "ietf-netconf-client" module in terms of its features and groupings.

2.1.1. Features

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

```
Features:
  +-- ssh-initiate
  +-- tls-initiate
  +-- ssh-listen
  +-- tls-listen
  +-- central-netconf-client-supported
```

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

2.1.2. Groupings

The "ietf-netconf-client" module defines the following "grouping" statements:

* netconf-client-initiate-stack-grouping
* netconf-client-listen-stack-grouping
* netconf-client-app-grouping

Each of these groupings are presented in the following subsections.

2.1.2.1. The "netconf-client-initiate-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-initiate-stack-grouping" grouping:

```
  grouping netconf-client-initiate-stack-grouping:
    +-- (transport)
       +--:(ssh) {ssh-initiate}?
       |  +-- ssh
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- ssh-client-parameters
       |     |  +---u sshc:ssh-client-grouping
       |     +-- netconf-client-parameters
       |        +--u ncc:netconf-client-grouping
       +--:(tls) {tls-initiate}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- netconf-client-parameters
                +---u ncc:netconf-client-grouping
```

Comments:

* The "netconf-client-initiate-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that initiate connections to NETCONF servers, as opposed to receiving call-home [RFC8071] connections.

* The "transport" choice node enables either the SSH or TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].

2.1.2.2. The "netconf-client-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-listen-stack-grouping" grouping:

```
  grouping netconf-client-listen-stack-grouping:
    +-- (transport)
       +--:(ssh) {ssh-listen}?
       |  +-- ssh
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- ssh-client-parameters
       |     |  +---u sshc:ssh-client-grouping
       |     +-- netconf-client-parameters
       |        +--u ncc:netconf-client-grouping
       +--:(tls) {tls-listen}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- netconf-client-parameters
                +---u ncc:netconf-client-grouping
```

Comments:

* The "netconf-client-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that receive call-home [RFC8071] connections from NETCONF servers.

* The "transport" choice node enables either the SSH or TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].

2.1.2.3. The "netconf-client-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-client-app-grouping" grouping:

```
  grouping netconf-client-app-grouping:
    +-- initiate! {ssh-initiate or tls-initiate}?
    |  +-- netconf-server* [name]
    |     +-- name?                 string
    |     +-- endpoints
    |     |  +-- endpoint* [name]
    |     |     +-- name?                                        string
    |     |     +---u netconf-client-initiate-stack-grouping
    |     +-- connection-type
    |     |  +-- (connection-type)
    |     |     +--:(persistent-connection)
    |     |     |  +-- persistent!
    |     |     +--:(periodic-connection)
    |     |        +-- periodic!
    |     |           +-- period?         uint16
    |     |           +-- anchor-time?    yang:date-and-time
    |     |           +-- idle-timeout?   uint16
    |     +-- reconnect-strategy
    |        +-- start-with?     enumeration
    |        +-- max-wait?       uint16
    |        +-- max-attempts?   uint8
    +-- listen! {ssh-listen or tls-listen}?
       +-- idle-timeout?   uint16
       +-- endpoints
          +-- endpoint* [name]
             +-- name?                                     string
             +---u netconf-client-listen-stack-grouping
```

Comments:

* The "netconf-client-app-grouping" defines the configuration for a NETCONF client that supports both initiating connections to NETCONF servers as well as receiving call-home connections from NETCONF servers.

* Both the "initiate" and "listen" subtrees are predicated by "feature" statements.

* For the referenced grouping statement(s):

  - The "netconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.1 in this document.
  - The "netconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.

2.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-netconf-client" module:

```
module: ietf-netconf-client
  +--rw netconf-client {central-netconf-client-supported}?
     +---u netconf-client-app-grouping
```

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 top-level node "netconf-client" is additionally constrained by the feature "central-netconf-client-supported".

* The "netconf-client-app-grouping" grouping is discussed in Section 2.1.2.3 in this document.

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

2.2. Example Usage

The following example illustrates configuring a NETCONF client to initiate connections, using both the SSH and TLS transport protocols, as well as to listen for call-home connections, again using both the SSH and TLS transport protocols.

This example is consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

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

<netconf-client xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-clie\
nt">

  <!-- NETCONF servers to initiate connections to -->
  <initiate>
    <netconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
```

```
            <name>corp-fw1.example.com</name>
            <ssh>
              <tcp-client-parameters>
                <remote-address>corp-fw1.example.com</remote-address>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <ssh-client-parameters>
                <client-identity>
                  <username>foobar</username>
                  <public-key>
                    <central-keystore-reference>ssh-rsa-key</central-k\
   eystore-reference>
                  </public-key>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <central-truststore-reference>trusted-server-ca-ce\
   rts</central-truststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <central-truststore-reference>trusted-server-ee-ce\
   rts</central-truststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <max-wait>30</max-wait>
                  <max-attempts>3</max-attempts>
                </keepalives>
              </ssh-client-parameters>
            </ssh>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <tls>
              <tcp-client-parameters>
                <remote-address>corp-fw2.example.com</remote-address>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
```

```
                  <certificate>
                    <central-keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </central-keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <central-truststore-reference>trusted-server-ca-ce\
rts</central-truststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <central-truststore-reference>trusted-server-ee-ce\
rts</central-truststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
            </tls>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
        </reconnect-strategy>
      </netconf-server>
    </initiate>

    <!-- endpoints to listen for NETCONF Call Home connections on -->
    <listen>
      <endpoints>
        <endpoint>
          <name>Intranet-facing SSH listener</name>
          <ssh>
            <tcp-server-parameters>
              <local-address>192.0.2.7</local-address>
            </tcp-server-parameters>
            <ssh-client-parameters>
              <client-identity>
```

```
                <username>foobar</username>
                <public-key>
                  <central-keystore-reference>ssh-rsa-key</central-key\
   store-reference>
                </public-key>
              </client-identity>
              <server-authentication>
                <ca-certs>
                  <central-truststore-reference>trusted-server-ca-cert\
   s</central-truststore-reference>
                </ca-certs>
                <ee-certs>
                  <central-truststore-reference>trusted-server-ee-cert\
   s</central-truststore-reference>
                </ee-certs>
                <ssh-host-keys>
                  <central-truststore-reference>trusted-ssh-public-key\
   s</central-truststore-reference>
                </ssh-host-keys>
              </server-authentication>
            </ssh-client-parameters>
          </ssh>
        </endpoint>
        <endpoint>
          <name>Intranet-facing TLS listener</name>
          <tls>
            <tcp-server-parameters>
              <local-address>192.0.2.7</local-address>
            </tcp-server-parameters>
            <tls-client-parameters>
              <client-identity>
                <certificate>
                  <central-keystore-reference>
                    <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                    <certificate>ex-rsa-cert</certificate>
                  </central-keystore-reference>
                </certificate>
              </client-identity>
              <server-authentication>
                <ca-certs>
                  <central-truststore-reference>trusted-server-ca-cert\
   s</central-truststore-reference>
                </ca-certs>
                <ee-certs>
                  <central-truststore-reference>trusted-server-ee-cert\
   s</central-truststore-reference>
                </ee-certs>
              </server-authentication>
```

```
              <keepalives>
                <peer-allowed-to-send/>
              </keepalives>
            </tls-client-parameters>
          </tls>
        </endpoint>
      </endpoints>
    </listen>
  </netconf-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7589], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

<CODE BEGINS> file "ietf-netconf-client@2024-03-16.yang"

```
module ietf-netconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix ncc;

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

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-ssh-client {
    prefix sshc;
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }
```

```
  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module contains a collection of YANG definitions
     for configuring NETCONF clients.

     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 HHHH
     (https://www.rfc-editor.org/info/rfcHHHH); 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 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC HHHH: NETCONF Client and Server Models";
  }

  // Features
```

```
    feature ssh-initiate {
      description
        "The 'ssh-initiate' feature indicates that the NETCONF client
         supports initiating SSH connections to NETCONF servers.";
      reference
        "RFC 6242:
           Using the NETCONF Protocol over Secure Shell (SSH)";
    }

    feature tls-initiate {
      description
        "The 'tls-initiate' feature indicates that the NETCONF client
         supports initiating TLS connections to NETCONF servers.";
      reference
        "RFC 7589: Using the NETCONF Protocol over Transport
           Layer Security (TLS) with Mutual X.509 Authentication";
    }

    feature ssh-listen {
      description
        "The 'ssh-listen' feature indicates that the NETCONF client
         supports opening a port to listen for incoming NETCONF
         server call-home SSH connections.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature tls-listen {
      description
        "The 'tls-listen' feature indicates that the NETCONF client
         supports opening a port to listen for incoming NETCONF
         server call-home TLS connections.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature central-netconf-client-supported {
      description
        "The 'central-netconf-client-supported' feature indicates
         that the server that implements this module supports
         the top-level 'netconf-client' node.

         This feature is needed as some servers may want to use
         features defined in this module, which requires this
         module to be implemented, without having to support
         the top-level 'netconf-client' node.";
    }
```

```
  // Groupings

  grouping netconf-client-grouping {
    description
      "A reusable grouping for configuring a NETCONF client
       without any consideration for how underlying transport
       sessions are established.

       This grouping currently does not define any nodes. It
       exists only so the model can be consistent with other
       'client-server' models.";
  }

  grouping netconf-client-initiate-stack-grouping {
    description
      "A reusable grouping for configuring a NETCONF client
       'initiate' protocol stack for a single outbound connection.";
    choice transport {
      mandatory true;
      description
        "Selects between available transports.";
      case ssh {
        if-feature "ssh-initiate";
        container ssh {
          description
            "Specifies TCP, SSH, and NETCONF configuration
             for the connection.";
          container tcp-client-parameters {
            description
              "TCP-level client parameters to initiate
               a NETCONF over SSH connection.";
            uses tcpc:tcp-client-grouping {
              refine "remote-port" {
                default "830";
                description
                  "The NETCONF client will attempt to connect
                   to the IANA-assigned well-known port value
                   for 'netconf-ssh' (830) if no value is
                   specified.";
              }
            }
          }
          container ssh-client-parameters {
            description
              "SSH-level client parameters to initiate
               a NETCONF over SSH connection.";
            uses sshc:ssh-client-grouping;
          }
```

```
            container netconf-client-parameters {
              description
                "NETCONF-level client parameters to initiate
                 a NETCONF over SSH connection.";
              uses ncc:netconf-client-grouping;
            }
          }
        }
        case tls {
          if-feature "tls-initiate";
          container tls {
            description
              "Specifies TCP, TLS, and NETCONF configuration
               for the connection.";
            container tcp-client-parameters {
              description
                "TCP-level client parameters to initiate
                 a NETCONF over TLS connection.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "6513";
                  description
                    "The NETCONF client will attempt to connect
                     to the IANA-assigned well-known port value
                     for 'netconf-tls' (6513) if no value is
                     specified.";
                }
              }
            }
            container tls-client-parameters {
              must client-identity {
                description
                  "NETCONF/TLS clients MUST pass some
                   authentication credentials.";
              }
              description
                "TLS-level client parameters to initiate
                 a NETCONF over TLS connection.";
              uses tlsc:tls-client-grouping;
            }
            container netconf-client-parameters {
              description
                "NETCONF-level client parameters to initiate
                 a NETCONF over TLS connection.";
              uses ncc:netconf-client-grouping;
            }
          }
        }
```

```
      }
    } // netconf-client-initiate-stack-grouping

    grouping netconf-client-listen-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         'listen' protocol stack for listening on a single port.  The
         'listen' stack supports call home connections, as
         described in RFC 8071";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-listen";
          container ssh {
            description
              "TCP, SSH, and NETCONF configuration to listen
               for NETCONF over SSH Call Home connections.";
            container tcp-server-parameters {
              description
                "TCP-level server parameters to listen for
                 NETCONF over SSH Call Home connections.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4334";
                  description
                    "The NETCONF client will listen on the IANA-
                     assigned well-known port for 'netconf-ch-ssh'
                     (4334) if no value is specified.";
                }
              }
            }
            container ssh-client-parameters {
              description
                "SSH-level client parameters to listen for
                 NETCONF over SSH Call Home connections.";
              uses sshc:ssh-client-grouping;
            }
            container netconf-client-parameters {
              description
                "NETCONF-level client parameters to listen for
                 NETCONF over SSH Call Home connections.";
              uses ncc:netconf-client-grouping;
            }
          }
```

```
        }
        case tls {
          if-feature "tls-listen";
          container tls {
            description
              "TCP, TLS, and NETCONF configuration to listen
               for NETCONF over TLS Call Home connections.";
            container tcp-server-parameters {
              description
                "TCP-level server parameters to listen for
                 NETCONF over TLS Call Home connections.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4335";
                  description
                    "The NETCONF client will listen on the IANA-
                     assigned well-known port for 'netconf-ch-tls'
                     (4335) if no value is specified.";
                }
              }
            }
            container tls-client-parameters {
              must client-identity {
                description
                  "NETCONF/TLS clients MUST pass some
                   authentication credentials.";
              }
              description
                "TLS-level client parameters to listen for
                 NETCONF over TLS Call Home connections.";
              uses tlsc:tls-client-grouping;
            }
            container netconf-client-parameters {
              description
                "NETCONF-level client parameters to listen for
                 NETCONF over TLS Call Home connections.";
              uses ncc:netconf-client-grouping;
            }
          }
        }
      }
    } // netconf-client-listen-stack-grouping

    grouping netconf-client-app-grouping {
      description
        "A reusable grouping for configuring a NETCONF client
         application that supports both 'initiate' and 'listen'
         protocol stacks for a multiplicity of connections.";
```

```
       container initiate {
         if-feature "ssh-initiate or tls-initiate";
         presence
           "Indicates that client-initiated connections have been
            configured.  This statement is present so the mandatory
            descendant nodes do not imply that this node must be
            configured.";
         description
           "Configures client initiating underlying TCP connections.";
         list netconf-server {
           key "name";
           min-elements 1;
           description
             "List of NETCONF servers the NETCONF client is to
              maintain simultaneous connections with.";
           leaf name {
             type string;
             description
               "An arbitrary name for the NETCONF server.";
           }
           container endpoints {
             description
               "Container for the list of endpoints.";
             list endpoint {
               key "name";
               min-elements 1;
               ordered-by user;
               description
                 "A user-ordered list of endpoints that the NETCONF
                  client will attempt to connect to in the specified
                  sequence.  Defining more than one enables
                  high-availability.";
               leaf name {
                 type string;
                 description
                   "An arbitrary name for the endpoint.";
               }
               uses netconf-client-initiate-stack-grouping;
             } // list endpoint
           } // container endpoints

           container connection-type {
             description
               "Indicates the NETCONF client's preference for how the
                NETCONF connection is maintained.";
             choice connection-type {
               mandatory true;
               description
```

```
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
                  presence
                    "Indicates that a persistent connection is to be
                     maintained.";
                  description
                    "Maintain a persistent connection to the NETCONF
                     server.  If the connection goes down, immediately
                     start trying to reconnect to the NETCONF server,
                     using the reconnection strategy.

                     This connection type minimizes any NETCONF server
                     to NETCONF client data-transfer delay, albeit at
                     the expense of holding resources longer.";
                }
              }
              case periodic-connection {
                container periodic {
                  presence "Indicates that a periodic connection is
                            to be maintained.";
                  description
                    "Periodically connect to the NETCONF server.

                     This connection type decreases resource
                     utilization, albeit with increased delay in
                     NETCONF server to NETCONF client interactions.

                     The NETCONF client should close the underlying
                     TCP connection upon completing planned activities.

                     Connections are established at the same start
                     time regardless how long the previous connection
                     stayed open.

                     In the case that the previous connection is still
                     active, establishing a new connection is NOT
                     RECOMMENDED.";
                  leaf period {
                    type uint16;
                    units "minutes";
                    default "60";
                    description
                      "Duration of time between periodic connections.";
                  }
                  leaf anchor-time {
                    type yang:date-and-time {
                      // constrained to minute-level granularity
```

```
                    pattern '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|[1-2]'
                          + '[0-9]|3[0-1])T(0[0-9]|1[0-9]|2[0-3]):['
                          + '0-5][0-9]:00(Z|[\+\-]((1[0-3]|0[0-9]):'
                          + '([0-5][0-9])|14:00))?';
                  }
                  description
                    "Designates a timestamp before or after which a
                     series of periodic connections are determined.
                     The periodic connections occur at a whole
                     multiple interval from the anchor time.

                     If an 'anchor-time' is not provided, then the
                     server may implicitly set it to the time when
                     this configuraton is applied (e.g., on boot).

                     For example, for an anchor time is 15 minutes
                     past midnight and a period interval of 24 hours,
                     then a periodic connection will occur 15 minutes
                     past midnight everyday.";
                }
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default 180; // three minutes
                  description
                    "Specifies the maximum number of seconds that
                     a NETCONF session may remain idle. A NETCONF
                     session will be dropped if it is idle for an
                     interval longer then this number of seconds.
                     If set to zero, then the NETCONF client will
                     never drop a session because it is idle.";
                }
              }
            }
          }
          container reconnect-strategy {
            description
              "The reconnection strategy directs how a NETCONF client
               reconnects to a NETCONF server, after discovering its
               connection to the server has dropped, even if due to a
               reboot.  The NETCONF client starts with the specified
               endpoint and tries to connect to it max-attempts times
               before trying the next endpoint in the list (round
               robin).";
            leaf start-with {
              type enumeration {
                enum first-listed {
```

```
                description
                  "Indicates that reconnections should start with
                   the first endpoint listed.";
              }
              enum last-connected {
                description
                  "Indicates that reconnections should start with
                   the endpoint last connected to.  If no previous
                   connection has ever been established, then the
                   first endpoint configured is used.   NETCONF
                   clients SHOULD be able to remember the last
                   endpoint connected to across reboots.";
              }
              enum random-selection {
                description
                  "Indicates that reconnections should start with
                   a random endpoint.";
              }
            }
            default "first-listed";
            description
              "Specifies which of the NETCONF server's endpoints
               the NETCONF client should start with when trying
               to connect to the NETCONF server.";
          }
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "5";
            description
              "Specifies the amount of time in seconds after which,
               if the connection is not established, an endpoint
               connection attempt is considered unsuccessful.";
          }
          leaf max-attempts {
            type uint8 {
              range "1..max";
            }
            default "3";
            description
              "Specifies the number times the NETCONF client tries
               to connect to a specific endpoint before moving on
               to the next endpoint in the list (round robin).";
          }
        }
      } // netconf-server
```

```
      } // initiate

      container listen {
        if-feature "ssh-listen or tls-listen";
        presence
          "Indicates that client-listening ports have been configured.
           This statement is present so the mandatory descendant nodes
           do not imply that this node must be configured.";
        description
          "Configures the client to accept call-home TCP connections.";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "180"; // three minutes
          description
            "Specifies the maximum number of seconds that a NETCONF
             session may remain idle. A NETCONF session will be
             dropped if it is idle for an interval longer than this
             number of seconds.  If set to zero, then the server
             will never drop a session because it is idle.";
        }
        container endpoints {
          description
            "Container for a list of endpoints.";
          list endpoint {
            key "name";
            min-elements 1;
            description
              "List of endpoints to listen for NETCONF connections.";
            leaf name {
              type string;
              description
                "An arbitrary name for the NETCONF listen endpoint.";
            }
            uses netconf-client-listen-stack-grouping;
          }
        }
      } // listen
    } // netconf-client-app-grouping

    // Protocol accessible node for clients that implement this module.
    container netconf-client {
      if-feature central-netconf-client-supported;
      uses netconf-client-app-grouping;
      description
        "Top-level container for NETCONF client configuration.";
    }
  }
```

<CODE ENDS>

3. The "ietf-netconf-server" Module

The NETCONF server model presented in this section supports both listening for connections as well as initiating call-home connections, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the NETCONF server supports.

3.1. Data Model Overview

This section provides an overview of the "ietf-netconf-server" module in terms of its features and groupings.

3.1.1. Features

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

```
Features:
  +-- ssh-listen
  +-- tls-listen
  +-- ssh-call-home
  +-- tls-call-home
  +-- central-netconf-server-supported
```

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

3.1.2. Groupings

The "ietf-netconf-server" module defines the following "grouping" statements:

* netconf-server-grouping
* netconf-server-listen-stack-grouping
* netconf-server-callhome-stack-grouping
* netconf-server-app-grouping

Each of these groupings are presented in the following subsections.

3.1.2.1. The "netconf-server-grouping" Grouping

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

```
  grouping netconf-server-grouping:
    +-- client-identity-mappings
       +---u x509c2n:cert-to-name
```

Comments:

* The "netconf-server-grouping" defines the configuration for the "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).

* The "client-identity-mappings" node defines a mapping from certificate fields to NETCONF user names.

* For the referenced grouping statement(s):

  - The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

3.1.2.2. The "netconf-server-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-listen-stack-grouping" grouping:

```
  grouping netconf-server-listen-stack-grouping:
    +-- (transport)
       +--:(ssh) {ssh-listen}?
       |  +-- ssh
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- ssh-server-parameters
       |     |  +---u sshs:ssh-server-grouping
       |     +-- netconf-server-parameters
       |        +---u ncs:netconf-server-grouping
       +--:(tls) {tls-listen}?
          +-- tls
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- netconf-server-parameters
                +---u ncs:netconf-server-grouping
```

Comments:

* The "netconf-server-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack for NETCONF servers that listen for connections from NETCONF clients, as opposed to initiating call-home [RFC8071] connections.

* The "transport" choice node enables either the SSH or TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.3. The "netconf-server-callhome-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-callhome-stack-grouping" grouping:

```
  grouping netconf-server-callhome-stack-grouping:
    +-- (transport)
       +--:(ssh) {ssh-call-home}?
       |  +-- ssh
       |     +-- tcp-client-parameters
       |     |  +---u tcpc:tcp-client-grouping
       |     +-- ssh-server-parameters
       |     |  +---u sshs:ssh-server-grouping
       |     +-- netconf-server-parameters
       |        +---u ncs:netconf-server-grouping
       +--:(tls) {tls-call-home}?
          +-- tls
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- netconf-server-parameters
                +---u ncs:netconf-server-grouping
```

Comments:

* The "netconf-server-callhome-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF servers that initiate call-home [RFC8071] connections to NETCONF clients.

* The "transport" choice node enables either the SSH or TLS transports to be configured, with each option enabled by a "feature" statement.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "ssh-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "netconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.4. The "netconf-server-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "netconf-server-app-grouping" grouping:

```
  grouping netconf-server-app-grouping:
    +-- listen! {ssh-listen or tls-listen}?
    |  +-- idle-timeout?   uint16
    |  +-- endpoints
    |     +-- endpoint* [name]
    |        +-- name?                                    string
    |        +---u netconf-server-listen-stack-grouping
    +-- call-home! {ssh-call-home or tls-call-home}?
       +-- netconf-client* [name]
          +-- name?                 string
          +-- endpoints
          |  +-- endpoint* [name]
          |     +-- name?                                       string
          |     +---u netconf-server-callhome-stack-grouping
          +-- connection-type
          |  +-- (connection-type)
          |     +--:(persistent-connection)
          |     |  +-- persistent!
          |     +--:(periodic-connection)
          |        +-- periodic!
          |           +-- period?         uint16
          |           +-- anchor-time?    yang:date-and-time
          |           +-- idle-timeout?   uint16
          +-- reconnect-strategy
             +-- start-with?     enumeration
             +-- max-wait?       uint16
             +-- max-attempts?   uint8
```

Comments:

* The "netconf-server-app-grouping" defines the configuration for a NETCONF server that supports both listening for connections from NETCONF clients as well as initiating call-home connections to NETCONF clients.

* Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "netconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
  - The "netconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-netconf-server" module:

```
module: ietf-netconf-server
  +--rw netconf-server {central-netconf-server-supported}?
     +---u netconf-server-app-grouping
```

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 top-level node "netconf-server" is additionally constrained by the feature "central-netconf-server-supported".

* The "netconf-server-app-grouping" grouping is discussed in Section 3.1.2.4 in this document.

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

3.2. Example Usage

The following example illustrates configuring a NETCONF server to listen for NETCONF client connections using both the SSH and TLS transport protocols, as well as configuring call-home to two NETCONF clients, one using SSH and the other using TLS.

This example is consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

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

<netconf-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server"
  xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">

  <!-- endpoints to listen for NETCONF connections on -->
  <listen>
    <endpoints>
      <endpoint> <!-- listening for SSH connections -->
        <name>netconf/ssh</name>
```

```
          <ssh>
            <tcp-server-parameters>
              <local-address>192.0.2.7</local-address>
            </tcp-server-parameters>
            <ssh-server-parameters>
              <server-identity>
                <host-key>
                  <name>deployment-specific-certificate</name>
                  <public-key>
                    <central-keystore-reference>ssh-rsa-key</central-k\
   eystore-reference>
                  </public-key>
                </host-key>
              </server-identity>
              <client-authentication>
              </client-authentication>
            </ssh-server-parameters>
            <netconf-server-parameters>
              <!-- nothing to configure -->
            </netconf-server-parameters>
          </ssh>
        </endpoint>
        <endpoint> <!-- listening for TLS sessions -->
          <name>netconf/tls</name>
          <tls>
            <tcp-server-parameters>
              <local-address>192.0.2.7</local-address>
            </tcp-server-parameters>
            <tls-server-parameters>
              <server-identity>
                <certificate>
                  <central-keystore-reference>
                    <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                    <certificate>ex-rsa-cert</certificate>
                  </central-keystore-reference>
                </certificate>
              </server-identity>
              <client-authentication>
                <ca-certs>
                  <central-truststore-reference>trusted-client-ca-cert\
   s</central-truststore-reference>
                </ca-certs>
                <ee-certs>
                  <central-truststore-reference>trusted-client-ee-cert\
   s</central-truststore-reference>
                </ee-certs>
              </client-authentication>
              <keepalives>
```

```
                <peer-allowed-to-send/>
              </keepalives>
            </tls-server-parameters>
            <netconf-server-parameters>
              <client-identity-mappings>
                <cert-to-name>
                  <id>1</id>
                  <fingerprint>11:0A:05:11:00</fingerprint>
                  <map-type>x509c2n:specified</map-type>
                  <name>scooby-doo</name>
                </cert-to-name>
                <cert-to-name>
                  <id>2</id>
                  <map-type>x509c2n:san-any</map-type>
                </cert-to-name>
              </client-identity-mappings>
            </netconf-server-parameters>
          </tls>
        </endpoint>
      </endpoints>
    </listen>

    <!-- calling home to SSH and TLS based NETCONF clients -->
    <call-home>
      <netconf-client> <!-- SSH-based client -->
        <name>config-mgr</name>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <ssh>
              <tcp-client-parameters>
                <remote-address>east.config-mgr.example.com</remote-ad\
dress>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <ssh-server-parameters>
                <server-identity>
                  <host-key>
                    <name>deployment-specific-certificate</name>
                    <public-key>
                      <central-keystore-reference>ssh-rsa-key</central\
-keystore-reference>
                    </public-key>
                  </host-key>
```

```
                </server-identity>
              </ssh-server-parameters>
              <netconf-server-parameters>
                <!-- nothing to configure -->
              </netconf-server-parameters>
            </ssh>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <ssh>
              <tcp-client-parameters>
                <remote-address>west.config-mgr.example.com</remote-ad\
   dress>
              </tcp-client-parameters>
              <ssh-server-parameters>
                <server-identity>
                  <host-key>
                    <name>deployment-specific-certificate</name>
                    <public-key>
                      <central-keystore-reference>ssh-rsa-key</central\
   -keystore-reference>
                    </public-key>
                  </host-key>
                </server-identity>
              </ssh-server-parameters>
              <netconf-server-parameters>
                <!-- nothing to configure -->
              </netconf-server-parameters>
            </ssh>
          </endpoint>
        </endpoints>
        <connection-type>
          <periodic>
            <idle-timeout>300</idle-timeout>
            <period>60</period>
          </periodic>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
          <max-wait>3</max-wait>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </netconf-client>
      <netconf-client> <!-- TLS-based client -->
        <name>data-collector</name>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
```

```
            <tls>
              <tcp-client-parameters>
                <remote-address>east.analytics.example.com</remote-add\
   ress>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <central-keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
   ey>
                      <certificate>ex-rsa-cert</certificate>
                    </central-keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <central-truststore-reference>trusted-client-ca-ce\
   rts</central-truststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <central-truststore-reference>trusted-client-ee-ce\
   rts</central-truststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <netconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
```

```
                  </cert-to-name>
                </client-identity-mappings>
              </netconf-server-parameters>
            </tls>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <tls>
              <tcp-client-parameters>
                <remote-address>west.analytics.example.com</remote-add\
ress>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <central-keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </central-keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <central-truststore-reference>trusted-client-ca-ce\
rts</central-truststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <central-truststore-reference>trusted-client-ee-ce\
rts</central-truststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <netconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
```

```
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </netconf-server-parameters>
            </tls>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
        <reconnect-strategy>
          <start-with>first-listed</start-with>
          <max-wait>3</max-wait>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </netconf-client>
    </call-home>
  </netconf-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7407], [RFC7589], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

<CODE BEGINS> file "ietf-netconf-server@2024-03-16.yang"

```
module ietf-netconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix ncs;

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

  import ietf-x509-cert-to-name {
    prefix x509c2n;
```

```
      reference
        "RFC 7407: A YANG Data Model for SNMP Configuration";
    }

    import ietf-tcp-client {
      prefix tcpc;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    import ietf-tcp-server {
      prefix tcps;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    import ietf-ssh-common {
      prefix sshcmn;
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    import ietf-ssh-server {
      prefix sshs;
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    import ietf-tls-server {
      prefix tlss;
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    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 <mailto:kent+ietf@watsen.net>";

    description
      "This module contains a collection of YANG definitions
       for configuring NETCONF servers.

       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 HHHH
        (https://www.rfc-editor.org/info/rfcHHHH); 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 2024-03-16 {
      description
        "Initial version";
      reference
        "RFC HHHH: NETCONF Client and Server Models";
    }

    // Features

    feature ssh-listen {
      description
        "The 'ssh-listen' feature indicates that the NETCONF server
         supports opening a port to accept NETCONF over SSH
         client connections.";
      reference
        "RFC 6242:
           Using the NETCONF Protocol over Secure Shell (SSH)";
    }

    feature tls-listen {
      description
        "The 'tls-listen' feature indicates that the NETCONF server
         supports opening a port to accept NETCONF over TLS
         client connections.";
      reference
        "RFC 7589: Using the NETCONF Protocol over Transport
                   Layer Security (TLS) with Mutual X.509
                   Authentication";
    }
```

```
    feature ssh-call-home {
      description
        "The 'ssh-call-home' feature indicates that the NETCONF
         server supports initiating a NETCONF over SSH call
         home connection to NETCONF clients.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature tls-call-home {
      description
        "The 'tls-call-home' feature indicates that the NETCONF
         server supports initiating a NETCONF over TLS call
         home connection to NETCONF clients.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature central-netconf-server-supported {
      description
        "The 'central-netconf-server-supported' feature indicates
         that the server supports the top-level 'netconf-server'
         node.

         This feature is needed as some servers may want to use
         features defined in this module, which requires this
         module to be implemented, without having to support
         the top-level 'netconf-server' node.";
    }

    // Groupings

    grouping netconf-server-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         without any consideration for how underlying transport
         sessions are established.

         Note that this grouping uses a fairly typical descendant
         node name such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue by wrapping the 'uses'
         statement in a container called, e.g.,
         'netconf-server-parameters'.  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container client-identity-mappings {
```

```
        description
          "Specifies mappings through which NETCONF client X.509
           certificates are used to determine a NETCONF username,
           per RFC 7407.

           For TLS-based transports, if no matching and valid
           cert-to-name list entry can be found, then the NETCONF
           server MUST close the connection, and MUST NOT accept
           NETCONF messages over it, per Section 7 in RFC 7589.

           For SSH-based transports, a matching cert-to-name
           entry overrides the username provided by the SSH
           implementation, consistent with the second paragraph
           of Section 3 in RFC 6242.";
        reference
          "RFC 6242:
             Using the NETCONF Protocol over Secure Shell (SSH)
           RFC 7589:
             Using the NETCONF Protocol over Transport Layer
             Security (TLS) with Mutual X.509 Authentication";
        uses x509c2n:cert-to-name {
          refine "cert-to-name/fingerprint" {
            mandatory false;
            description
              "A 'fingerprint' value does not need to be specified
               when the 'cert-to-name' mapping is independent of
               fingerprint matching.  A 'cert-to-name' having no
               fingerprint value will match any client certificate
               and therefore should only be present at the end of
               the user-ordered 'cert-to-name' list.";
          }
        }
      }
    }

    grouping netconf-server-listen-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         'listen' protocol stack for listening on a single port.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-listen";
          container ssh {
            description
              "TCP, SSH, and NETCONF configuration to listen
```

```
           for NETCONF over SSH connections.";
         container tcp-server-parameters {
           description
             "TCP-level server parameters to listen
              for NETCONF over SSH connections.";
           uses tcps:tcp-server-grouping {
             refine "local-port" {
               default "830";
               description
                 "The NETCONF server will listen on the
                  IANA-assigned well-known port value
                  for 'netconf-ssh' (830) if no value
                  is specified.";
             }
           }
         }
         container ssh-server-parameters {
           description
             "SSH-level server parameters to listen
              for NETCONF over SSH connections.";
           uses sshs:ssh-server-grouping;
         }
         container netconf-server-parameters {
           description
             "NETCONF-level server parameters to listen
              for NETCONF over SSH connections.";
           uses ncs:netconf-server-grouping {
             refine "client-identity-mappings" {
               if-feature "sshcmn:ssh-x509-certs";
               description
                 "Adds in an 'if-feature' statement
                  ensuring the 'client-identity-mappings'
                  descendant is enabled only when SSH
                  supports X.509 certificates.";
             }
             augment "client-identity-mappings" {
               description
                 "Adds a flag indicating if a cert-to-name
                  is required.";
               leaf mapping-required {
                 type boolean;
                 description
                   "Indicates that the cert-to-name mapping
                    is required (i.e., the SSH-level username
                    is ignored).";
               }
             }
           }
```

```
            }
          }
        }
        case tls {
          if-feature "tls-listen";
          container tls {
            description
              "TCP, TLS, and NETCONF configuration to listen
               for NETCONF over TLS connections.";
            container tcp-server-parameters {
              description
                "TCP-level server parameters to listen
                 for NETCONF over TLS connections.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "6513";
                  description
                    "The NETCONF server will listen on the
                     IANA-assigned well-known port value
                     for 'netconf-tls' (6513) if no value
                     is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "TLS-level server parameters to listen
                 for NETCONF over TLS connections.";
              uses tlss:tls-server-grouping {
                refine "client-authentication" {
                  must 'ca-certs or ee-certs';
                  description
                    "NETCONF/TLS servers MUST validate client
                     certificates.  This configures certificates
                     at the socket-level (i.e. bags).  More
                     discriminating client-certificate checks
                     SHOULD be implemented by the application.";
                  reference
                    "RFC 7589:
                      Using the NETCONF Protocol over Transport Layer
                      Security (TLS) with Mutual X.509 Authentication";
                }
              }
            }
            container netconf-server-parameters {
              description
                "NETCONF-level server parameters to listen
                 for NETCONF over TLS connections.";
```

```
              uses ncs:netconf-server-grouping {
                refine "client-identity-mappings/cert-to-name" {
                  min-elements 1;
                  description
                    "The TLS transport requires a mapping.";
                }
              }
            }
          }
        }
      }
    }

    grouping netconf-server-callhome-stack-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         'call-home' protocol stack, for a single outbound
         connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case ssh {
          if-feature "ssh-call-home";
          container ssh {
            description
              "TCP, SSH, and NETCONF configuration to initiate
               a NETCONF over SSH Call Home connection.";
            container tcp-client-parameters {
              description
                "TCP-level client parameters to initiate a
                 NETCONF over SSH Call Home connection.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "4334";
                  description
                    "The NETCONF server will attempt to connect
                     to the IANA-assigned well-known port for
                     'netconf-ch-ssh' (4334) if no value is
                     specified.";
                }
              }
            }
            container ssh-server-parameters {
              description
                "SSH-level server parameters to initiate a
                 NETCONF over SSH Call Home connection.";
              uses sshs:ssh-server-grouping;
```

```
            }
            container netconf-server-parameters {
              description
                "NETCONF-level server parameters to initiate a
                 NETCONF over SSH Call Home connection.";
              uses ncs:netconf-server-grouping {
                refine "client-identity-mappings" {
                  if-feature "sshcmn:ssh-x509-certs";
                  description
                    "Adds in an 'if-feature' statement
                     ensuring the 'client-identity-mappings'
                     descendant is enabled only when SSH
                     supports X.509 certificates.";
                }
                augment "client-identity-mappings" {
                  description
                    "Adds a flag indicating if a cert-to-name
                     is required.";
                  leaf mapping-required {
                    type boolean;
                    description
                      "Indicates that the cert-to-name mapping
                       is required (i.e., the SSH-level username
                       is ignored).";
                  }
                }
              }
            }
          }
          case tls {
            if-feature "tls-call-home";
            container tls {
              description
                "TCP, TLS, and NETCONF configuration to initiate
                 a NETCONF over TLS Call Home connection.";
              container tcp-client-parameters {
                description
                  "TCP-level client parameters to initiate a
                   NETCONF over TLS Call Home connection.";
                uses tcpc:tcp-client-grouping {
                  refine "remote-port" {
                    default "4335";
                    description
                      "The NETCONF server will attempt to connect
                       to the IANA-assigned well-known port for
                       'netconf-ch-tls' (4335) if no value is
                       specified.";
```

```
              }
            }
          }
          container tls-server-parameters {
            description
              "TLS-level server parameters to initiate a
               NETCONF over TLS Call Home connection.";
            uses tlss:tls-server-grouping {
              refine "client-authentication" {
                must 'ca-certs or ee-certs';
                description
                  "NETCONF/TLS servers MUST validate client
                   certificates.  This configures certificates
                   at the socket-level (i.e. bags).  More
                   discriminating client-certificate checks
                   SHOULD be implemented by the application.";
                reference
                  "RFC 7589:
                    Using the NETCONF Protocol over Transport Layer
                    Security (TLS) with Mutual X.509 Authentication";
              }
            }
          }
          container netconf-server-parameters {
            description
              "NETCONF-level server parameters to initiate a
               NETCONF over TLS Call Home connection.";
            uses ncs:netconf-server-grouping {
              refine "client-identity-mappings/cert-to-name" {
                min-elements 1;
                description
                  "The TLS transport requires a mapping.";
              }
            }
          }
        }
      }
    }

    grouping netconf-server-app-grouping {
      description
        "A reusable grouping for configuring a NETCONF server
         application that supports both 'listen' and 'call-home'
         protocol stacks for a multiplicity of connections.";
      container listen {
        if-feature "ssh-listen or tls-listen";
        presence
```

```
          "Indicates that server-listening ports have been configured.
           This statement is present so the mandatory descendant
           nodes do not imply that this node must be configured.";
        description
          "Configures listen behavior";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "180"; // three minutes
          description
            "Specifies the maximum number of seconds that a NETCONF
             session may remain idle. A NETCONF session will be
             dropped if it is idle for an interval longer than this
             number of seconds.  If set to zero, then the server
             will never drop a session because it is idle.";
        }
        container endpoints {
          description
            "Container for a list of endpoints.";
          list endpoint {
            key "name";
            min-elements 1;
            description
              "List of endpoints to listen for NETCONF connections.";
            leaf name {
              type string;
              description
                "An arbitrary name for the NETCONF listen endpoint.";
            }
            uses netconf-server-listen-stack-grouping;
          }
        }
      }
      container call-home {
        if-feature "ssh-call-home or tls-call-home";
        presence
          "Indicates that server-initiated call home connections have
           been configured.  This statement is present so the mandatory
           descendant nodes do not imply that this node must be
           configured.";
        description
          "Configures the NETCONF server to initiate the underlying
           transport connection to NETCONF clients.";
        list netconf-client {
          key "name";
          min-elements 1;
          description
            "List of NETCONF clients the NETCONF server is to
```

```
           maintain simultaneous call-home connections with.";
         leaf name {
           type string;
           description
             "An arbitrary name for the remote NETCONF client.";
         }
         container endpoints {
           description
             "Container for the list of endpoints.";
           list endpoint {
             key "name";
             min-elements 1;
             ordered-by user;
             description
               "A non-empty user-ordered list of endpoints for this
                NETCONF server to try to connect to in sequence.
                Defining more than one enables high-availability.";
             leaf name {
               type string;
               description
                 "An arbitrary name for this endpoint.";
             }
             uses netconf-server-callhome-stack-grouping;
           }
         }
         container connection-type {
           description
             "Indicates the NETCONF server's preference for how the
              NETCONF connection is maintained.";
           choice connection-type {
             mandatory true;
             description
               "Selects between available connection types.";
             case persistent-connection {
               container persistent {
                 presence
                   "Indicates that a persistent connection is to be
                    maintained.";
                 description
                   "Maintain a persistent connection to the NETCONF
                    client. If the connection goes down, immediately
                    start trying to reconnect to the NETCONF client,
                    using the reconnection strategy.

                    This connection type minimizes any NETCONF client
                    to NETCONF server data-transfer delay, albeit at
                    the expense of holding resources longer.";
               }
```

```
            }
            case periodic-connection {
              container periodic {
                presence "Indicates that a periodic connection is
                          to be maintained.";
                description
                  "Periodically connect to the NETCONF client.

                   This connection type decreases resource
                   utilization, albeit with increased delay in
                   NETCONF client to NETCONF server interactions.

                   The NETCONF client SHOULD gracefully close the
                   connection using <close-session> upon completing
                   planned activities.  If the NETCONF session is
                   not closed gracefully, the NETCONF server MUST
                   immediately attempt to reestablish the connection.

                   Connections are established at the same start
                   time regardless how long the previous connection
                   stayed open.

                   In the case that the previous connection is still
                   active (i.e., the NETCONF client has not closed
                   it yet), establishing a new connection is NOT
                   RECOMMENDED.";
                leaf period {
                  type uint16;
                  units "minutes";
                  default "60";
                  description
                    "Duration of time between periodic connections.";
                }
                leaf anchor-time {
                  type yang:date-and-time {
                    // constrained to minute-level granularity
                    pattern '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|[1-2]'
                          + '[0-9]|3[0-1])T(0[0-9]|1[0-9]|2[0-3]):['
                          + '0-5][0-9]:00(Z|[\+\-]((1[0-3]|0[0-9]):'
                          + '([0-5][0-9])|14:00))?';
                  }
                  description
                    "Designates a timestamp before or after which a
                     series of periodic connections are determined.
                     The periodic connections occur at a whole
                     multiple interval from the anchor time.

                     If an 'anchor-time' is not provided, then the
```

```
                    server may implicitly set it to the time when
                    this configuraton is applied (e.g., on boot).

                    For example, for an anchor time is 15 minutes
                    past midnight and a period interval of 24 hours,
                    then a periodic connection will occur 15 minutes
                    past midnight everyday.";
                }
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default "180"; // three minutes
                  description
                    "Specifies the maximum number of seconds that
                     a NETCONF session may remain idle. A NETCONF
                     session will be dropped if it is idle for an
                     interval longer than this number of seconds.
                     If set to zero, then the server will never
                     drop a session because it is idle.";
                }
              }
            } // case periodic-connection
          } // choice connection-type
        } // container connection-type
        container reconnect-strategy {
          description
            "The reconnection strategy directs how a NETCONF server
             reconnects to a NETCONF client, after discovering its
             connection to the client has dropped, even if due to a
             reboot.  The NETCONF server starts with the specified
             endpoint and tries to connect to it max-attempts times
             before trying the next endpoint in the list (round
             robin).";
          leaf start-with {
            type enumeration {
              enum first-listed {
                description
                  "Indicates that reconnections should start with
                   the first endpoint listed.";
              }
              enum last-connected {
                description
                  "Indicates that reconnections should start with
                   the endpoint last connected to.  If no previous
                   connection has ever been established, then the
                   first endpoint configured is used.   NETCONF
                   servers SHOULD be able to remember the last
                   endpoint connected to across reboots.";
```

```
                }
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default "first-listed";
              description
                "Specifies which of the NETCONF client's endpoints
                 the NETCONF server should start with when trying
                 to connect to the NETCONF client.";
            }
            leaf max-wait {
              type uint16 {
                range "1..max";
              }
              units "seconds";
              default "5";
              description
                "Specifies the amount of time in seconds after which,
                 if the connection is not established, an endpoint
                 connection attempt is considered unsuccessful.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the NETCONF server tries
                 to connect to a specific endpoint before moving on
                 to the next endpoint in the list (round robin).";
            }
          } // container reconnect-strategy
        } // list netconf-client
      } // container call-home
    } // grouping netconf-server-app-grouping

    // Protocol accessible node for servers that implement this module.
    container netconf-server {
      if-feature central-netconf-server-supported;
      uses netconf-server-app-grouping;
      description
        "Top-level container for NETCONF server configuration.";
    }
  }
```

<CODE ENDS>

4. Security Considerations

4.1. Considerations for the "ietf-netconf-client" YANG Module

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

The "ietf-netconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

4.2. Considerations for the "ietf-netconf-server" YANG Module

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

The "ietf-netconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-netconf-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-client
prefix:       ncc
reference:    RFC HHHH

name:         ietf-netconf-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-server
prefix:       ncs
reference:    RFC HHHH
```

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[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>.

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-05, 21 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

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

[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>.

[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>.

[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>.

[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>.

Appendix A. Change Log

A.1. 00 to 01

* Renamed "keychain" to "keystore".

A.2. 01 to 02

* Added to ietf-netconf-client ability to connected to a cluster of endpoints, including a reconnection-strategy.

* Added to ietf-netconf-client the ability to configure connection-type and also keep-alive strategy.

* Updated both modules to accommodate new groupings in the ssh/tls drafts.

A.3. 02 to 03

* Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

* Changed 'netconf-client' to be a grouping (not a container).

A.4. 03 to 04

* Added RFC 8174 to Requirements Language Section.

* Replaced refine statement in ietf-netconf-client to add a mandatory true.

* Added refine statement in ietf-netconf-server to add a must statement.

* Now there are containers and groupings, for both the client and server models.

A.5. 04 to 05

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated examples to inline key and certificates (no longer a leafref to keystore)

A.6. 05 to 06

* Fixed change log missing section issue.

* Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.

* Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

* Removed "idle-timeout" from "persistent" connection config.

* Added "random-selection" for reconnection-strategy's "starts-with" enum.

* Replaced "connection-type" choice default (persistent) with "mandatory true".

* Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.

* Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

* Modified examples to be compatible with new crypto-types algs

A.9. 08 to 09

* Corrected use of "mandatory true" for "address" leafs.

* Updated examples to reflect update to groupings defined in the keystore draft.

* Updated to use groupings defined in new TCP and HTTP drafts.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

* Reformatted YANG modules.

A.11. 10 to 11

* Adjusted for the top-level "demux container" added to groupings imported from other modules.

* Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Moved "expanded" tree diagrams to the Appendix.

A.12. 11 to 12

* Removed the "Design Considerations" section.

* Removed the 'must' statement limiting keepalives in periodic connections.

* Updated models and examples to reflect removal of the "demux" containers in the imported models.

* Updated the "periodic-connnection" description statements to be more like the RESTCONF draft, especially where it described dropping the underlying TCP connection.

* Updated text to better reference where certain examples come from (e.g., which Section in which draft).

* In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.

* Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

A.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

A.14. 13 to 14

* Adjusting from change in TLS client model (removing the top-level 'certificate' container), by swapping refining-in a 'mandatory true' statement with a 'must' statement outside the 'uses' statement.

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

A.15. 14 to 15

* Refactored both the client and server modules similar to how the ietf-restconf-server module was refactored in -13 of that draft, and the ietf-restconf-client grouping.

A.16. 15 to 16

* Added refinement to make "cert-to-name/fingerprint" be mandatory false.

* Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

A.17. 16 to 17

* Updated examples to include the "*-key-format" nodes.

* Updated examples to remove the "required" nodes.

* Updated examples to remove the "client-auth-defined-elsewhere" nodes.

A.18. 17 to 18

* Updated examples to reflect new "bag" addition to truststore.

A.19. 18 to 19

* Updated examples to remove the 'algorithm' nodes.
* Updated examples to reflect the new TLS keepalives structure.
* Added keepalives to the tcp-client-parameters section in the netconf-server SSH-based call-home example.
* Added a TLS-based call-home example to the netconf-client example.
* Added a "Note to Reviewers" note to first page.

A.20. 19 to 20

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
* Removed expanded tree diagrams that were listed in the Appendix.
* Updated the Security Considerations section.

A.21. 20 to 21

* Cleaned up titles in the IANA Considerations section
* Fixed issues found by the SecDir review of the "keystore" draft.

A.22. 21 to 22

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.23. 22 to 23

* Floated an 'if-feature' statement in a grouping down to where the grouping is used.
* Clarified 'client-identity-mappings' for both the SSH and TLS transports.
* For netconf-client, augmented-in a 'mapping-required' flag into 'client-identity-mappings' only for the SSH transport, and refined-in a 'min-elements 1' only for the TLS transport.

* Aligned modules with ‘pyang -f‘ formatting.

A.24. 23 to 24

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

* Minor editorial nits

A.25. 24 to 25

* Fixed up the ’WG Web’ and ’WG List’ lines in YANG module(s)

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

A.26. 25 to 26

* Added feature "central-netconf-client-supported" to top-level node "netconf-client".

* Added feature "central-netconf-server-supported" to top-level node "netconf-server".

* Clarified container "netconf-client-parameters" description statement.

* Removed unneccesary "xmlns:x509c2n" in a NETCONF server configuration example.

A.27. 26 to 27

* Updated per Shepherd reviews impacting the suite of drafts.

* Added "max-wait" leaf to the "reconnect-strategy" nodes.

A.28. 27 to 28

* Updated per Shepherd reviews impacting the suite of drafts.

A.29. 28 to 29

* Updated (implicitly) via Tom Petch reviews.

* Fixed pattern statement for "leaf anchor-time".

A.30. 29 to 30

* Addresses AD review comments.

* Added note to Editor to fix line foldings.

* Removed netconf-client-grouping, since it was empty.

* Removed erronious statement "client-identity-mappings" must be enabled by a "feature".

* Added Security Considerations text to also look a SC-section from imported modules.

* Removed "A wrapper around the foobar parameters to avoid name collisions" text.

* Added container "endpoints" to wrap list "endpoint".

A.31. 30 to 31

* Addresses AD review by Rob Wilton.

A.32. 31 to 32

* Addresses 1st-round of IESG reviews.

A.33. 32 to 34

* Addresses issues found in OpsDir review of the ssh-client-server draft.

* s/defines/presents/ in a few places.

* Add refs to where the 'operational' and 'system' datastores are defined.

* Renamed Security Considerations section s/Template for/ Considerations for/

A.34. 34 to 36

* Nothing changed. Only bumped for automation...

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balázs Kovács, Benoit Claise, Bert Wijnen, David Lamparter, Jürgen Schönwälder, Ladislav Lhotka, Martin Björklund, Mehmet Ersue, Michal Vako, Phil Shafer, Qiufang Ma, Radek Krejci, Ramkumar Dhanapal, Rob Wilton, Sean Turner, and Tom Petch.

Author's Address

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

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 January 2021

K. Watsen
Watsen Networks

8 July 2020

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-20

Abstract

This document defines two YANG modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server. Both modules support the TLS transport protocol with both standard RESTCONF and RESTCONF Call Home connections.

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 (note: not all may be present):

* "AAAA" --> the assigned RFC value for draft-ietf-netconf-crypto-types
* "BBBB" --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* "CCCC" --> the assigned RFC value for draft-ietf-netconf-keystore
* "DDDD" --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* "EEEE" --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
* "FFFF" --> the assigned RFC value for draft-ietf-netconf-tls-client-server
* "GGGG" --> the assigned RFC value for draft-ietf-netconf-http-client-server

* "HHHH" --> the assigned RFC value for draft-ietf-netconf-netconf-client-server

* "IIII" --> the assigned RFC value for this draft

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

* "2020-07-08" --> the publication date of this draft

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

* Appendix B. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

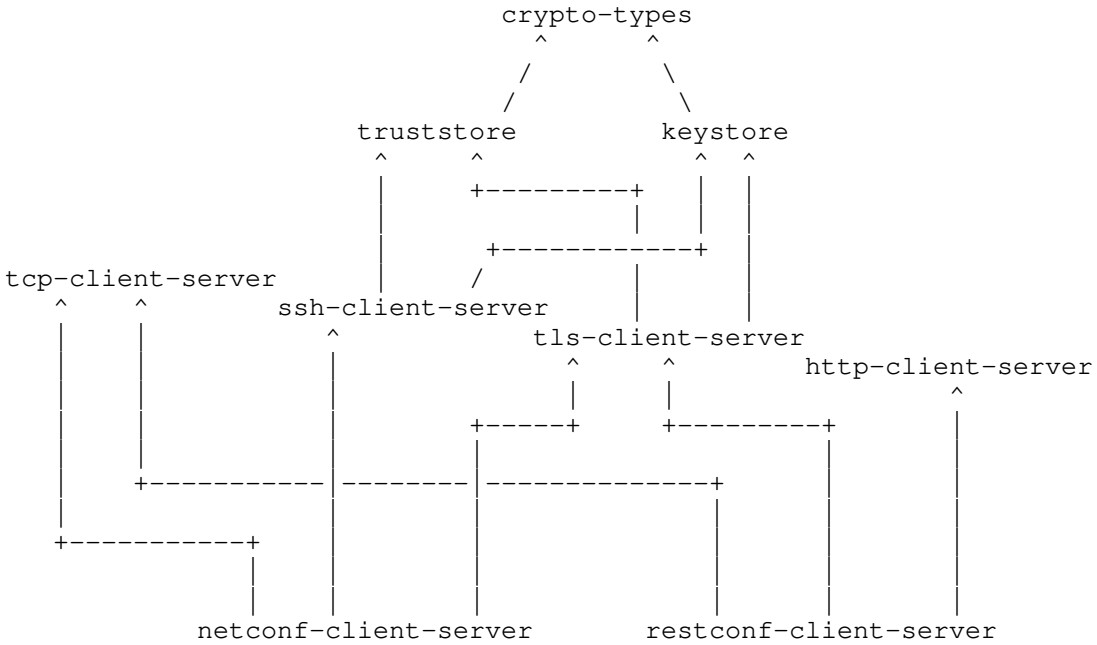
This document defines two YANG [RFC7950] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server [RFC8040]. Both modules support the TLS [RFC8446] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [RFC8071].

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

2. The "ietf-restconf-client" Module

The RESTCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the RESTCONF client supports.

2.1. Data Model Overview

2.1.1. Features

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

```
Features:
  +-- https-initiate
  +-- http-listen
  +-- https-listen
```

2.1.2. Groupings

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

```
Groupings:
  +-- restconf-client-grouping
  +-- restconf-client-initiate-stack-grouping
  +-- restconf-client-listen-stack-grouping
  +-- restconf-client-app-grouping
```

Each of these groupings are presented in the following subsections.

2.1.2.1. The "restconf-client-grouping" Grouping

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

```
  grouping restconf-client-grouping ---> <empty>
```

Comments:

* This grouping does not define any nodes, but is maintained so that downstream modules can augment nodes into it if needed.

* The "restconf-client-grouping" defines, if it can be called that, the configuration for just "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

2.1.2.2. The "restconf-client-initiate-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-initiate-stack-grouping" grouping:

```
  grouping restconf-client-initiate-stack-grouping
    +-- (transport)
       +--:(https) {https-initiate}?
          +-- https
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
             |  +---u httpc:http-client-grouping
             +-- restconf-client-parameters
                +---u rcc:restconf-client-grouping
```

Comments:

* The "restconf-client-initiate-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that initiate connections to RESTCONF servers, as opposed to receiving call-home [RFC8071] connections.

* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.3. The "restconf-client-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-listen-stack-grouping" grouping:

```
  grouping restconf-client-listen-stack-grouping
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-client-parameters
       |     |  +---u httpc:http-client-grouping
       |     +-- restconf-client-parameters
       |        +---u rcc:restconf-client-grouping
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
             |  +---u httpc:http-client-grouping
             +-- restconf-client-parameters
                +---u rcc:restconf-client-grouping
```

Comments:

* The "restconf-client-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that receive call-home [RFC8071] connections from RESTCONF servers.

* The "transport" choice node enables both the HTTP and HTTPS transports to be configured, with each option enabled by a "feature" statement. Note that RESTCONF requires HTTPS, the HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-client.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.4. The "restconf-client-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-app-grouping" grouping:

```
  grouping restconf-client-app-grouping
    +-- initiate! {https-initiate}?
    |  +-- restconf-server* [name]
    |     +-- name?                 string
    |     +-- endpoints
    |     |  +-- endpoint* [name]
    |     |     +-- name?                                          string
    |     |     +---u restconf-client-initiate-stack-grouping
    |     +-- connection-type
    |     |  +-- (connection-type)
    |     |     +--:(persistent-connection)
    |     |     |  +-- persistent!
    |     |     +--:(periodic-connection)
    |     |        +-- periodic!
    |     |           +-- period?         uint16
    |     |           +-- anchor-time?    yang:date-and-time
    |     |           +-- idle-timeout?   uint16
    |     +-- reconnect-strategy
    |        +-- start-with?     enumeration
    |        +-- max-attempts?   uint8
    +-- listen! {http-listen or https-listen}?
       +-- idle-timeout?   uint16
       +-- endpoint* [name]
          +-- name?                                       string
          +---u restconf-client-listen-stack-grouping
```

Comments:

* The "restconf-client-app-grouping" defines the configuration for a RESTCONF client that supports both initiating connections to RESTCONF servers as well as receiving call-home connections from RESTCONF servers.

* Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "restconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.
  - The "restconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

2.1.3. Protocol-accessible Nodes

The following diagram lists all the protocol-accessible nodes defined in the "ietf-restconf-client" module:

```
module: ietf-restconf-client
  +--rw restconf-client
     +---u restconf-client-app-grouping
```

Comments:

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

* For the "ietf-restconf-client" module, the protocol-accessible nodes are an instance of the "restconf-client-app-grouping" discussed in Section 2.1.2.4 grouping.

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

2.2. Example Usage

The following example illustrates configuring a RESTCONF client to initiate connections, as well as to listen for call-home connections.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

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

  <!-- RESTCONF servers to initiate connections to -->
  <initiate>
    <restconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <https>
            <tcp-client-parameters>
```

```
                <remote-address>corp-fw1.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <client-identity>
                  <basic>
                    <user-id>bob</user-id>
                    <password>secret</password>
                  </basic>
                </client-identity>
              </http-client-parameters>
            </https>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <https>
              <tcp-client-parameters>
```

```
                <remote-address>corp-fw2.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <client-identity>
                  <basic>
                    <user-id>bob</user-id>
                    <password>secret</password>
                  </basic>
                </client-identity>
              </http-client-parameters>
            </https>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
```

```
      </restconf-server>
    </initiate>

    <!-- endpoints to listen for RESTCONF Call Home connections on -->
    <listen>
      <endpoint>
        <name>Intranet-facing listener</name>
        <https>
          <tcp-server-parameters>
            <local-address>11.22.33.44</local-address>
          </tcp-server-parameters>
          <tls-client-parameters>
            <client-identity>
              <certificate>
                <keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </keystore-reference>
              </certificate>
            </client-identity>
            <server-authentication>
              <ca-certs>
                <truststore-reference>trusted-server-ca-certs</truststo\
  re-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-server-ee-certs</truststo\
  re-reference>
              </ee-certs>
            </server-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-client-parameters>
          <http-client-parameters>
            <client-identity>
              <basic>
                <user-id>bob</user-id>
                <password>secret</password>
              </basic>
            </client-identity>
          </http-client-parameters>
        </https>
      </endpoint>
    </listen>
  </restconf-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC8040], and [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-tls-client-server], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-client@2020-07-08.yang"

```
module ietf-restconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-client";
  prefix rcc;

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

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-http-client {
    prefix httpc;
    reference
      "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

     description
       "This module contains a collection of YANG definitions
        for configuring RESTCONF clients.

        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 IIII
        (https://www.rfc-editor.org/info/rfcIIII); see the RFC
        itself for full legal notices.

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

     revision 2020-07-08 {
       description
         "Initial version";
       reference
         "RFC IIII: RESTCONF Client and Server Models";
     }

     // Features

     feature https-initiate {
       description
         "The 'https-initiate' feature indicates that the RESTCONF
          client supports initiating HTTPS connections to RESTCONF
          servers. This feature exists as HTTPS might not be a
          mandatory to implement transport in the future.";
       reference
         "RFC 8040: RESTCONF Protocol";
     }
```

```
    feature http-listen {
      description
        "The 'https-listen' feature indicates that the RESTCONF client
         supports opening a port to listen for incoming RESTCONF
         server call-home connections.  This feature exists as not
         all RESTCONF clients may support RESTCONF call home.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature https-listen {
      description
        "The 'https-listen' feature indicates that the RESTCONF client
         supports opening a port to listen for incoming RESTCONF
         server call-home connections.  This feature exists as not
         all RESTCONF clients may support RESTCONF call home.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    // Groupings

    grouping restconf-client-grouping {
      description
        "A reusable grouping for configuring a RESTCONF client
         without any consideration for how underlying transport
         sessions are established.

         This grouping currently doesn't define any nodes.";
    }

    grouping restconf-client-initiate-stack-grouping {
      description
        "A reusable grouping for configuring a RESTCONF client
         'initiate' protocol stack for a single connection.";

      choice transport {
        mandatory true;
        description
          "Selects between available transports. This is a
           'choice' statement so as to support additional
           transport options to be augmented in.";
        case https {
          if-feature "https-initiate";
          container https {
            must 'tls-client-parameters/client-identity
                  or http-client-parameters/client-identity';
            description
```

```
            "Specifies HTTPS-specific transport
             configuration.";
          container tcp-client-parameters {
            description
              "A wrapper around the TCP client parameters
               to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
              refine "remote-port" {
                default "443";
                description
                  "The RESTCONF client will attempt to
                   connect to the IANA-assigned well-known
                   port value for 'https' (443) if no value
                   is specified.";
              }
            }
          }
          container tls-client-parameters {
            description
              "A wrapper around the TLS client parameters
               to avoid name collisions.";
            uses tlsc:tls-client-grouping;
          }
          container http-client-parameters {
            description
              "A wrapper around the HTTP client parameters
               to avoid name collisions.";
            uses httpc:http-client-grouping;
          }
          container restconf-client-parameters {
            description
              "A wrapper around the HTTP client parameters
               to avoid name collisions.";
            uses rcc:restconf-client-grouping;
          }
        }
      }
    }
  } // restconf-client-initiate-stack-grouping

  grouping restconf-client-listen-stack-grouping {
    description
      "A reusable grouping for configuring a RESTCONF client
       'listen' protocol stack for a single connection.  The
       'listen' stack supports call home connections, as
       described in RFC 8071";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
```

```
      choice transport {
        mandatory true;
        description
          "Selects between available transports. This is a
           'choice' statement so as to support additional
           transport options to be augmented in.";
        case http {
          if-feature "http-listen";
          container http {
            description
              "HTTP-specific listening configuration for inbound
               connections.

               This transport option is made available to support
               deployments where the TLS connections are terminated
               by another system (e.g., a load balanacer) fronting
               the client.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4336";
                  description
                    "The RESTCONF client will listen on the IANA-
                     assigned well-known port for 'restconf-ch-tls'
                     (4336) if no value is specified.";
                }
              }
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP client parameters
                 to avoid name collisions.";
              uses httpc:http-client-grouping;
            }
            container restconf-client-parameters {
              description
                "A wrapper around the RESTCONF client parameters
                 to avoid name collisions.";
              uses rcc:restconf-client-grouping;
            }
          }
        }
        case https {
          if-feature "https-listen";
          container https {
```

```
            must 'tls-client-parameters/client-identity
                  or http-client-parameters/client-identity';
            description
              "HTTPS-specific listening configuration for inbound
               connections.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "4336";
                  description
                    "The RESTCONF client will listen on the IANA-
                     assigned well-known port for 'restconf-ch-tls'
                     (4336) if no value is specified.";
                }
              }
            }
            container tls-client-parameters {
              description
                "A wrapper around the TLS client parameters
                 to avoid name collisions.";
              uses tlsc:tls-client-grouping;
            }
            container http-client-parameters {
              description
                "A wrapper around the HTTP client parameters
                 to avoid name collisions.";
              uses httpc:http-client-grouping;
            }
            container restconf-client-parameters {
              description
                "A wrapper around the RESTCONF client parameters
                 to avoid name collisions.";
              uses rcc:restconf-client-grouping;
            }
          }
        }
      }
    } // restconf-client-listen-stack-grouping

    grouping restconf-client-app-grouping {
      description
        "A reusable grouping for configuring a RESTCONF client
         application that supports both 'initiate' and 'listen'
         protocol stacks for a multiplicity of connections.";
      container initiate {
```

```
        if-feature "https-initiate";
        presence "Enables client to initiate TCP connections";
        description
          "Configures client initiating underlying TCP connections.";
        list restconf-server {
          key "name";
          min-elements 1;
          description
            "List of RESTCONF servers the RESTCONF client is to
             maintain simultaneous connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF server.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
              description
                "A non-empty user-ordered list of endpoints for this
                 RESTCONF client to try to connect to in sequence.
                 Defining more than one enables high-availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this endpoint.";
              }
              uses restconf-client-initiate-stack-grouping;
            }
          }
          container connection-type {
            description
              "Indicates the RESTCONF client's preference for how
               the RESTCONF connection is maintained.";
            choice connection-type {
              mandatory true;
              description
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
                  presence "Indicates that a persistent connection
                            is to be maintained.";
                  description
                    "Maintain a persistent connection to the
```

```
                    RESTCONF server. If the connection goes down,
                    immediately start trying to reconnect to the
                    RESTCONF server, using the reconnection strategy.

                    This connection type minimizes any RESTCONF server
                    to RESTCONF client data-transfer delay, albeit
                    at the expense of holding resources longer.";
                }
              }
              case periodic-connection {
                container periodic {
                  presence "Indicates that a periodic connection is
                            to be maintained.";
                  description
                    "Periodically connect to the RESTCONF server.

                     This connection type increases resource
                     utilization, albeit with increased delay
                     in RESTCONF server to RESTCONF client
                     interactions.

                     The RESTCONF client SHOULD gracefully close
                     the underlying TLS connection upon completing
                     planned activities.

                     In the case that the previous connection is
                     still active, establishing a new connection
                     is NOT RECOMMENDED.";
                  leaf period {
                    type uint16;
                    units "minutes";
                    default "60";
                    description
                      "Duration of time between periodic
                       connections.";
                  }
                  leaf anchor-time {
                    type yang:date-and-time {
                      // constrained to minute-level granularity
                      pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                            + '(Z|[\+\-]\d{2}:\d{2})';
                    }
                    description
                      "Designates a timestamp before or after which
                       a series of periodic connections are
                       determined.  The periodic connections occur
                       at a whole multiple interval from the anchor
                       time.  For example, for an anchor time is 15
```

```
                    minutes past midnight and a period interval
                    of 24 hours, then a periodic connection will
                    occur 15 minutes past midnight everyday.";
                }
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default 120; // two minutes
                  description
                    "Specifies the maximum number of seconds
                     that the underlying TCP session may remain
                     idle. A TCP session will be dropped if it
                     is idle for an interval longer than this
                     number of seconds If set to zero, then the
                     RESTCONF client will never drop a session
                     because it is idle.";
                }
              }
            } // periodic-connection
          } // connection-type
        } // connection-type
        container reconnect-strategy {
          description
            "The reconnection strategy directs how a RESTCONF
             client reconnects to a RESTCONF server, after
             discovering its connection to the server has
             dropped, even if due to a reboot.  The RESTCONF
             client starts with the specified endpoint and
             tries to connect to it max-attempts times before
             trying the next endpoint in the list (round
             robin).";
          leaf start-with {
            type enumeration {
              enum first-listed {
                description
                  "Indicates that reconnections should start
                   with the first endpoint listed.";
              }
              enum last-connected {
                description
                  "Indicates that reconnections should start
                   with the endpoint last connected to.  If
                   no previous connection has ever been
                   established, then the first endpoint
                   configured is used.   RESTCONF clients
                   SHOULD be able to remember the last
                   endpoint connected to across reboots.";
              }
```

```
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default "first-listed";
              description
                "Specifies which of the RESTCONF server's
                 endpoints the RESTCONF client should start
                 with when trying to connect to the RESTCONF
                 server.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the RESTCONF client
                 tries to connect to a specific endpoint before
                 moving on to the next endpoint in the list
                 (round robin).";
            }
          }
        }
      } // initiate
      container listen {
        if-feature "http-listen or https-listen";
        presence "Enables client to accept call-home connections";
        description
          "Configures the client to accept call-home TCP connections.";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default 3600; // one hour
          description
            "Specifies the maximum number of seconds that an
             underlying TCP session may remain idle. A TCP session
             will be dropped if it is idle for an interval longer
             then this number of seconds.  If set to zero, then
             the server will never drop a session because it is
             idle.  Sessions that have a notification subscription
             active are never dropped.";
        }
        list endpoint {
          key "name";
          min-elements 1;
```

```
          description
            "List of endpoints to listen for RESTCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF listen endpoint.";
          }
          uses restconf-client-listen-stack-grouping;
        }
      }
    } // restconf-client-app-grouping

    // Protocol accessible node, for servers that implement
    // this module.
    container restconf-client {
      uses restconf-client-app-grouping;
      description
        "Top-level container for RESTCONF client configuration.";
    }
  }
```

<CODE ENDS>

3. The "ietf-restconf-server" Module

The RESTCONF server model presented in this section supports both listening for connections as well as initiating call-home connections.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the RESTCONF server supports.

3.1. Data Model Overview

3.1.1. Features

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

```
Features:
  +-- http-listen
  +-- https-listen
  +-- https-call-home
```

3.1.2. Groupings

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

```
Groupings:
  +-- restconf-server-grouping
  +-- restconf-server-listen-stack-grouping
  +-- restconf-server-callhome-stack-grouping
  +-- restconf-server-app-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "restconf-server-grouping" Grouping

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

```
  grouping restconf-server-grouping
    +-- client-identity-mappings
       +---u x509c2n:cert-to-name
```

Comments:

* The "restconf-server-grouping" defines the configuration for just "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).

* The "client-identity-mappings" node, which must be enabled by "feature" statements, defines a mapping from certificate fields to RESTCONF user names.

* For the referenced grouping statement(s):

  - The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

3.1.2.2. The "restconf-server-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-listen-stack-grouping" grouping:

```
  grouping restconf-server-listen-stack-grouping
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- external-endpoint!
       |     |  +-- address    inet:ip-address
       |     |  +-- port?      inet:port-number
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-server-parameters
       |     |  +---u https:http-server-grouping
       |     +-- restconf-server-parameters
       |        +---u rcs:restconf-server-grouping
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
             |  +---u https:http-server-grouping
             +-- restconf-server-parameters
                +---u rcs:restconf-server-grouping
```

Comments:

* The "restconf-server-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack for RESTCONF servers that listen for standard connections from RESTCONF clients, as opposed to initiating call-home [RFC8071] connections.

* The "transport" choice node enables both the HTTP and HTTPS transports to be configured, with each option enabled by a "feature" statement. The HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-server.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.3. The "restconf-server-callhome-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-callhome-stack-grouping" grouping:

```
  grouping restconf-server-callhome-stack-grouping
    +-- (transport)
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
             |  +---u https:http-server-grouping
             +-- restconf-server-parameters
                +---u rcs:restconf-server-grouping
```

Comments:

* The "restconf-server-callhome-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF servers that initiate call-home [RFC8071] connections to RESTCONF clients.

* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.4. The "restconf-server-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-app-grouping" grouping:

```
grouping restconf-server-app-grouping
  +-- listen! {http-listen or https-listen}?
  |  +-- endpoint* [name]
  |     +-- name?                                      string
  |     +---u restconf-server-listen-stack-grouping
  +-- call-home! {https-call-home}?
     +-- restconf-client* [name]
        +-- name?                 string
        +-- endpoints
        |  +-- endpoint* [name]
        |     +-- name?                                        string
        |     +---u restconf-server-callhome-stack-grouping
        +-- connection-type
        |  +-- (connection-type)
        |     +--:(persistent-connection)
        |     |  +-- persistent!
        |     +--:(periodic-connection)
        |        +-- periodic!
        |           +-- period?         uint16
        |           +-- anchor-time?    yang:date-and-time
        |           +-- idle-timeout?   uint16
        +-- reconnect-strategy
           +-- start-with?     enumeration
           +-- max-attempts?   uint8
```

Comments:

* The "restconf-server-app-grouping" defines the configuration for a RESTCONF server that supports both listening for connections from RESTCONF clients as well as initiatiating call-home connections to RESTCONF clients.

* Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "restconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
  - The "restconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following diagram lists all the protocol-accessible nodes defined in the "ietf-restconf-server" module:

```
module: ietf-restconf-server
  +--rw restconf-server
     +---u restconf-server-app-grouping
```

Comments:

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

* For the "ietf-restconf-server" module, the protocol-accessible nodes are an instance of the "restconf-server-app-grouping" discussed in Section 3.1.2.4 grouping.

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

3.2. Example Usage

The following example illustrates configuring a RESTCONF server to listen for RESTCONF client connections, as well as configuring call-home to one RESTCONF client.

This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-trust-anchors] and Section 2.2 of [I-D.ietf-netconf-keystore].

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

<restconf-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">

  <!-- endpoints to listen for RESTCONF connections on -->
  <listen>
    <endpoint>
      <name>restconf/https</name>
      <https>
        <tcp-server-parameters>
          <local-address>11.22.33.44</local-address>
        </tcp-server-parameters>
        <tls-server-parameters>
          <server-identity>
            <certificate>
              <keystore-reference>
```

```
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </keystore-reference>
              </certificate>
            </server-identity>
            <client-authentication>
              <ca-certs>
                <truststore-reference>trusted-client-ca-certs</truststo\
re-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-client-ee-certs</truststo\
re-reference>
              </ee-certs>
            </client-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-server-parameters>
          <http-server-parameters>
            <server-name>foo.example.com</server-name>
          </http-server-parameters>
          <restconf-server-parameters>
            <client-identity-mappings>
              <cert-to-name>
                <id>1</id>
                <fingerprint>11:0A:05:11:00</fingerprint>
                <map-type>x509c2n:specified</map-type>
                <name>scooby-doo</name>
              </cert-to-name>
                <cert-to-name>
                <id>2</id>
                <map-type>x509c2n:san-any</map-type>
              </cert-to-name>
            </client-identity-mappings>
          </restconf-server-parameters>
        </https>
      </endpoint>
    </listen>

    <!-- call home to a RESTCONF client with two endpoints -->
    <call-home>
      <restconf-client>
        <name>config-manager</name>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <https>
```

```
              <tcp-client-parameters>
                <remote-address>east.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
   ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
   ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
   ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <http-server-parameters>
                <server-name>foo.example.com</server-name>
              </http-server-parameters>
              <restconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
```

```
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </restconf-server-parameters>
            </https>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <https>
              <tcp-client-parameters>
                <remote-address>west.example.com</remote-address>
                <keepalives>
                  <idle-time>15</idle-time>
                  <max-probes>3</max-probes>
                  <probe-interval>30</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <http-server-parameters>
                <server-name>foo.example.com</server-name>
              </http-server-parameters>
              <restconf-server-parameters>
```

```
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </restconf-server-parameters>
            </https>
          </endpoint>
        </endpoints>
        <connection-type>
          <periodic>
            <idle-timeout>300</idle-timeout>
            <period>60</period>
          </periodic>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </restconf-client>
    </call-home>
  </restconf-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC7407], [RFC8040], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-tls-client-server], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-server@2020-07-08.yang"

```
module ietf-restconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-server";
  prefix rcs;

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

```
  }

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

  import ietf-x509-cert-to-name {
    prefix x509c2n;
    reference
      "RFC 7407: A YANG Data Model for SNMP Configuration";
  }

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-server {
    prefix tlss;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-http-server {
    prefix https;
    reference
      "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
     Author:   Gary Wu <mailto:garywu@cisco.com>
     Author:   Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>";
```

```
  description
    "This module contains a collection of YANG definitions
     for configuring RESTCONF servers.

     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 IIII
     (https://www.rfc-editor.org/info/rfcIIII); see the RFC
     itself for full legal notices.

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

  revision 2020-07-08 {
    description
      "Initial version";
    reference
      "RFC IIII: RESTCONF Client and Server Models";
  }

  // Features

  feature http-listen {
    description
      "The 'http-listen' feature indicates that the RESTCONF server
       supports opening a port to listen for incoming RESTCONF over
       TPC client connections, whereby the TLS connections are
       terminated by an external system.";
    reference
      "RFC 8040: RESTCONF Protocol";
  }

  feature https-listen {
    description
      "The 'https-listen' feature indicates that the RESTCONF server
       supports opening a port to listen for incoming RESTCONF over
```

```
          TLS client connections, whereby the TLS connections are
          terminated by the server itself.";
        reference
          "RFC 8040: RESTCONF Protocol";
      }

      feature https-call-home {
        description
          "The 'https-call-home' feature indicates that the RESTCONF
           server supports initiating connections to RESTCONF clients.";
        reference
          "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
      }


      // Groupings

      grouping restconf-server-grouping {
        description
          "A reusable grouping for configuring a RESTCONF server
           without any consideration for how underlying transport
           sessions are established.

           Note that this grouping uses a fairly typical descendent
           node name such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue by wrapping the 'uses'
           statement in a container called, e.g.,
           'restconf-server-parameters'.  This model purposely does
           not do this itself so as to provide maximum flexibility
           to consuming models.";

        container client-identity-mappings {
          description
            "Specifies mappings through which RESTCONF client X.509
             certificates are used to determine a RESTCONF username.
             If no matching and valid cert-to-name list entry can be
             found, then the RESTCONF server MUST close the connection,
             and MUST NOT accept RESTCONF messages over it.";
          reference
            "RFC 7407: A YANG Data Model for SNMP Configuration.";
          uses x509c2n:cert-to-name {
            refine "cert-to-name/fingerprint" {
              mandatory false;
              description
                "A 'fingerprint' value does not need to be specified
                 when the 'cert-to-name' mapping is independent of
                 fingerprint matching.  A 'cert-to-name' having no
```

```
                fingerprint value will match any client certificate
                and therefore should only be present at the end of
                the user-ordered 'cert-to-name' list.";
            }
          }
        }
      }

      grouping restconf-server-listen-stack-grouping {
        description
          "A reusable grouping for configuring a RESTCONF server
           'listen' protocol stack for a single connection.";
        choice transport {
          mandatory true;
          description
            "Selects between available transports. This is a
             'choice' statement so as to support additional
             transport options to be augmented in.";
          case http {
            if-feature "http-listen";
            container http {
              description
                "Configures RESTCONF server stack assuming that
                 TLS-termination is handled externally.";
              container external-endpoint {
                presence
                  "Specifies configuration for an external endpoint.";
                description
                  "Identifies contact information for the external
                   system that terminates connections before passing
                   them thru to this server (e.g., a network address
                   translator or a load balancer).  These values have
                   no effect on the local operation of this server, but
                   may be used by the application when needing to
                   inform other systems how to contact this server.";
                leaf address {
                  type inet:ip-address;
                  mandatory true;
                  description
                    "The IP address or hostname of the external system
                     that terminates incoming RESTCONF client
                     connections before forwarding them to this
                     server.";
                }
                leaf port {
                  type inet:port-number;
                  default "443";
                  description
```

```
                "The port number that the external system listens
                 on for incoming RESTCONF client connections that
                 are forwarded to this server.  The default HTTPS
                 port (443) is used, as expected for a RESTCONF
                 connection.";
            }
          }
          container tcp-server-parameters {
            description
              "A wrapper around the TCP server parameters
               to avoid name collisions.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "80";
                description
                  "The RESTCONF server will listen on the IANA-
                   assigned well-known port value for 'http'
                   (80) if no value is specified.";
              }
            }
          }
          container http-server-parameters {
            description
              "A wrapper around the HTTP server parameters
               to avoid name collisions.";
            uses https:http-server-grouping;
          }
          container restconf-server-parameters {
            description
              "A wrapper around the RESTCONF server parameters
               to avoid name collisions.";
            uses rcs:restconf-server-grouping;
          }
        }
      }
      case https {
        if-feature "https-listen";
        container https {
          description
            "Configures RESTCONF server stack assuming that
             TLS-termination is handled internally.";
          container tcp-server-parameters {
            description
              "A wrapper around the TCP server parameters
               to avoid name collisions.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "443";
```

```
                  description
                    "The RESTCONF server will listen on the IANA-
                     assigned well-known port value for 'https'
                     (443) if no value is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "A wrapper around the TLS server parameters
                 to avoid name collisions.";
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP server parameters
                 to avoid name collisions.";
              uses https:http-server-grouping;
            }
            container restconf-server-parameters {
              description
                "A wrapper around the RESTCONF server parameters
                 to avoid name collisions.";
              uses rcs:restconf-server-grouping;
            }
          }
        }
      }
    }

    grouping restconf-server-callhome-stack-grouping {
      description
        "A reusable grouping for configuring a RESTCONF server
         'call-home' protocol stack, for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports. This is a
           'choice' statement so as to support additional
           transport options to be augmented in.";
        case https {
          if-feature "https-listen";
          container https {
            description
              "Configures RESTCONF server stack assuming that
               TLS-termination is handled internally.";
            container tcp-client-parameters {
              description
```

```
                "A wrapper around the TCP client parameters
                 to avoid name collisions.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "4336";
                  description
                    "The RESTCONF server will attempt to
                     connect to the IANA-assigned well-known
                     port for 'restconf-ch-tls' (4336) if no
                     value is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "A wrapper around the TLS server parameters
                 to avoid name collisions.";
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP server parameters
                 to avoid name collisions.";
              uses https:http-server-grouping;
            }
            container restconf-server-parameters {
              description
                "A wrapper around the RESTCONF server parameters
                 to avoid name collisions.";
              uses rcs:restconf-server-grouping;
            }
          }
        }
      }
    }


    grouping restconf-server-app-grouping {
      description
        "A reusable grouping for configuring a RESTCONF server
         application that supports both 'listen' and 'call-home'
         protocol stacks for a multiplicity of connections.";
      container listen {
        if-feature "http-listen or https-listen";
        presence
          "Enables the RESTCONF server to listen for RESTCONF
           client connections.";
        description "Configures listen behavior";
```

```
        list endpoint {
          key "name";
          min-elements 1;
          description
            "List of endpoints to listen for RESTCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF listen endpoint.";
          }
          uses restconf-server-listen-stack-grouping;
        }
      }
      container call-home {
        if-feature "https-call-home";
        presence
          "Enables the RESTCONF server to initiate the underlying
           transport connection to RESTCONF clients.";
        description "Configures call-home behavior";
        list restconf-client {
          key "name";
          min-elements 1;
          description
            "List of RESTCONF clients the RESTCONF server is to
             maintain simultaneous call-home connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the remote RESTCONF client.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
              description
                "User-ordered list of endpoints for this RESTCONF
                 client.  Defining more than one enables high-
                 availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this endpoint.";
              }
              uses restconf-server-callhome-stack-grouping;
            }
```

```
          }
          container connection-type {
            description
              "Indicates the RESTCONF server's preference for how the
               RESTCONF connection is maintained.";
            choice connection-type {
              mandatory true;
              description
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
                  presence "Indicates that a persistent connection is
                            to be maintained.";
                  description
                    "Maintain a persistent connection to the RESTCONF
                     client. If the connection goes down, immediately
                     start trying to reconnect to the RESTCONF server,
                     using the reconnection strategy.

                     This connection type minimizes any RESTCONF
                     client to RESTCONF server data-transfer delay,
                     albeit at the expense of holding resources
                     longer.";
                }
              }
              case periodic-connection {
                container periodic {
                  presence "Indicates that a periodic connection is
                            to be maintained.";
                  description
                    "Periodically connect to the RESTCONF client.

                     This connection type increases resource
                     utilization, albeit with increased delay in
                     RESTCONF client to RESTCONF client interactions.

                     The RESTCONF client SHOULD gracefully close
                     the underlying TLS connection upon completing
                     planned activities.  If the underlying TLS
                     connection is not closed gracefully, the
                     RESTCONF server MUST immediately attempt
                     to reestablish the connection.

                     In the case that the previous connection is
                     still active (i.e., the RESTCONF client has not
                     closed it yet), establishing a new connection
                     is NOT RECOMMENDED.";
```

```
                leaf period {
                  type uint16;
                  units "minutes";
                  default "60";
                  description
                    "Duration of time between periodic connections.";
                }
                leaf anchor-time {
                  type yang:date-and-time {
                    // constrained to minute-level granularity
                    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                          + '(Z|[\+\-]\d{2}:\d{2})';
                  }
                  description
                    "Designates a timestamp before or after which a
                     series of periodic connections are determined.
                     The periodic connections occur at a whole
                     multiple interval from the anchor time.  For
                     example, for an anchor time is 15 minutes past
                     midnight and a period interval of 24 hours, then
                     a periodic connection will occur 15 minutes past
                     midnight everyday.";
                }
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default 120; // two minutes
                  description
                    "Specifies the maximum number of seconds that
                     the underlying TCP session may remain idle.
                     A TCP session will be dropped if it is idle
                     for an interval longer than this number of
                     seconds.  If set to zero, then the server
                     will never drop a session because it is idle.";
                }
              }
            }
          }
        }
        container reconnect-strategy {
          description
            "The reconnection strategy directs how a RESTCONF server
             reconnects to a RESTCONF client after discovering its
             connection to the client has dropped, even if due to a
             reboot.  The RESTCONF server starts with the specified
             endpoint and tries to connect to it max-attempts times
             before trying the next endpoint in the list (round
             robin).";
```

```
            leaf start-with {
              type enumeration {
                enum first-listed {
                  description
                    "Indicates that reconnections should start with
                     the first endpoint listed.";
                }
                enum last-connected {
                  description
                    "Indicates that reconnections should start with
                     the endpoint last connected to.  If no previous
                     connection has ever been established, then the
                     first endpoint configured is used.   RESTCONF
                     servers SHOULD be able to remember the last
                     endpoint connected to across reboots.";
                }
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default "first-listed";
              description
                "Specifies which of the RESTCONF client's endpoints
                 the RESTCONF server should start with when trying
                 to connect to the RESTCONF client.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the RESTCONF server tries
                 to connect to a specific endpoint before moving on to
                 the next endpoint in the list (round robin).";
            }
          }
        } // restconf-client
      } // call-home
    } // restconf-server-app-grouping



    // Protocol accessible node, for servers that implement
    // this module.
    container restconf-server {
```

```
      uses restconf-server-app-grouping;
      description
        "Top-level container for RESTCONF server configuration.";
    }

  }

  <CODE ENDS>
```

4. Security Considerations

4.1. The "ietf-restconf-client" YANG Module

The "ietf-restconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

4.2. The "ietf-restconf-server" YANG Module

The "ietf-restconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

5. IANA Considerations

5.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-restconf-client
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-server
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 two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registrations are requested:

```
      name:         ietf-restconf-client
      namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-client
      prefix:       ncc
      reference:    RFC IIII

      name:         ietf-restconf-server
      namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-server
      prefix:       ncs
      reference:    RFC IIII
```

6. References

6.1. Normative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

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

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-

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

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

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

[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>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

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

[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>.

[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>.

Appendix A. Expanded Tree Diagrams

A.1. Expanded Tree Diagram for 'ietf-restconf-client'

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-restconf-client" module.

This tree diagram shows all the nodes defined in this module, including those defined by "grouping" statements used by this module.

Please see Section 2.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

XNSERT_TEXT_FROM_FILE(refs/ietf-restconf-client-tree.txt)

A.2. Expanded Tree Diagram for 'ietf-restconf-server'

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-restconf-server" module.

This tree diagram shows all the nodes defined in this module, including those defined by "grouping" statements used by this module.

Please see Section 3.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

XNSERT_TEXT_FROM_FILE(refs/ietf-restconf-server-tree.txt)

Appendix B. Change Log

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

B.1. 00 to 01

* Renamed "keychain" to "keystore".

B.2. 01 to 02

* Filled in previously missing 'ietf-restconf-client' module.

* Updated the ietf-restconf-server module to accommodate new grouping 'ietf-tls-server-grouping'.

B.3. 02 to 03

* Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

* Changed restconf-client??? to be a grouping (not a container).

B.4. 03 to 04

* Added RFC 8174 to Requirements Language Section.
* Replaced refine statement in ietf-restconf-client to add a mandatory true.
* Added refine statement in ietf-restconf-server to add a must statement.
* Now there are containers and groupings, for both the client and server models.
* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Updated examples to inline key and certificates (no longer a leafref to keystore)

B.5. 04 to 05

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Updated examples to inline key and certificates (no longer a leafref to keystore)

B.6. 05 to 06

* Fixed change log missing section issue.
* Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.
* Reduced line length of the YANG modules to fit within 69 columns.

B.7. 06 to 07

* removed "idle-timeout" from "persistent" connection config.
* Added "random-selection" for reconnection-strategy's "starts-with" enum.
* Replaced "connection-type" choice default (persistent) with "mandatory true".
* Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.
* Replaced reconnect-timeout with period/anchor-time combo.

B.8. 07 to 08

* Modified examples to be compatible with new crypto-types algs

B.9. 08 to 09

* Corrected use of "mandatory true" for "address" leafs.

* Updated examples to reflect update to groupings defined in the keystore draft.

* Updated to use groupings defined in new TCP and HTTP drafts.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

* Reformatted YANG modules.

B.11. 10 to 11

* Adjusted for the top-level "demux container" added to groupings imported from other modules.

* Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Moved "expanded" tree diagrams to the Appendix.

B.12. 11 to 12

* Removed the 'must' statement limiting keepalives in periodic connections.

* Updated models and examples to reflect removal of the "demux" containers in the imported models.

* Updated the "periodic-connnection" description statements to better describe behavior when connections are not closed gracefully.

* Updated text to better reference where certain examples come from (e.g., which Section in which draft).

* In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.

* Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

B.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

* In ietf-restconf-server, Added 'http-listen' (not https-listen) choice, to support case when server is behind a TLS-terminator.

* Refactored server module to be more like other 'server' models. If folks like it, will also apply to the client model, as well as to both the netconf client/server models. Now the 'restconf-server-grouping' is just the RC-specific bits (i.e., the "demux" container minus the container), 'restconf-server-[listen|callhome]-stack-grouping' is the protocol stack for a single connection, and 'restconf-server-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

B.14. 13 to 14

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

* Adjusting from change in TLS client model (removing the top-level 'certificate' container).

* Added "external-endpoint" to the "http-listen" choice in ietf-restconf-server.

B.15. 14 to 15

* Added missing "or https-listen" clause in a "must" expression.

* Refactored the client module similar to how the server module was refactored in -13. Now the 'restconf-client-grouping' is just the RC-specific bits, the 'restconf-client-[initiate|listen]-stack-grouping' is the protocol stack for a single connection, and 'restconf-client-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

B.16. 15 to 16

* Added refinement to make "cert-to-name/fingerprint" be mandatory false.

* Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

* Updated restconf-client example to reflect that http-client-grouping no longer has a "protocol-version" leaf.

B.17. 16 to 17

* Updated examples to include the "*-key-format" nodes.

* Updated examples to remove the "required" nodes.

B.18. 17 to 18

* Updated examples to reflect new "bag" addition to truststore.

B.19. 18 to 19

* Updated examples to remove the 'algorithm' nodes.

* Updated examples to reflect the new TLS keepalives structure.

* Removed the 'protocol-versions' node from the restconf-server examples.

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

B.20. 19 to 20

* Moved and changed "must" statement so that either TLS *or* HTTP auth must be configured.

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

* Updated the Security Considerations section.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Mehmet Ersue, Ramkumar Dhanapal, Balazs Kovacs, Radek Krejci, David Lamparter, Ladislav Lhotka, Alan Luchuk, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Bert Wijnen.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-36

Abstract

This document presents two YANG modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server. Both modules support the TLS transport protocol with both standard RESTCONF and RESTCONF Call Home connections.

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 (note: not all may be present):

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* EEEE --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
* FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
* GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server

* HHHH --> the assigned RFC value for draft-ietf-netconf-netconf-client-server

* IIII --> 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:

* 2024-03-16 --> the publication date of this draft

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

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A. Change Log

Status of This Memo

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

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

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

This Internet-Draft will expire on 17 September 2024.

Table of Contents

1. Introduction

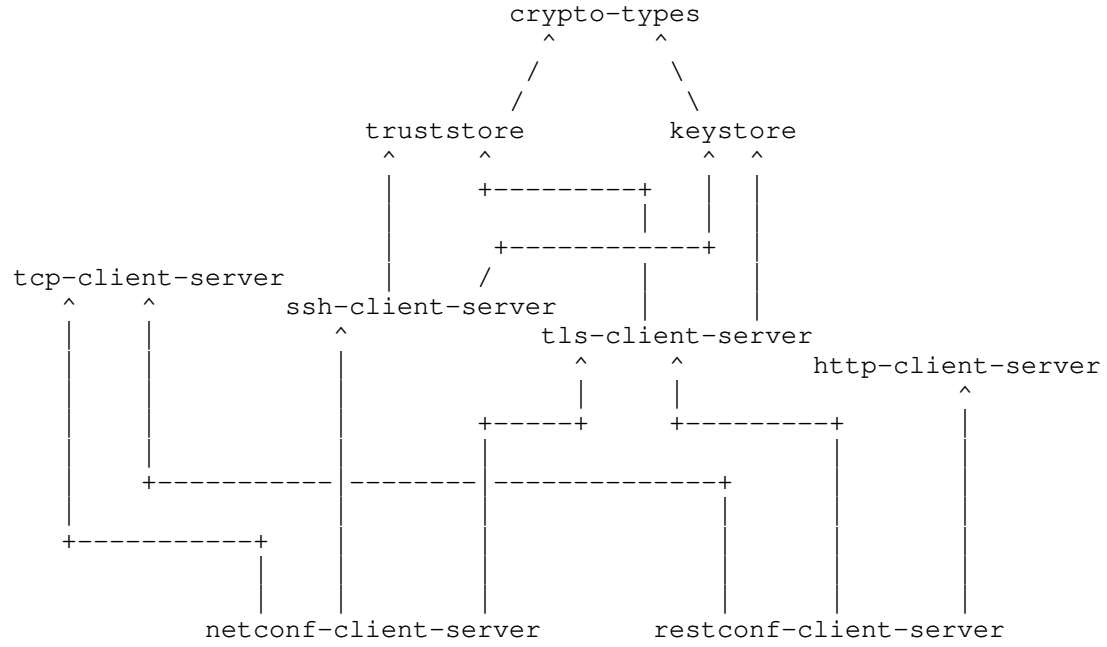
This document presents two YANG [RFC7950] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server [RFC8040]. Both modules support the TLS [RFC8446] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [RFC8071].

1.1. Relation to other RFCs

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

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



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

Table 1: Label 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, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational> (Section 5.3 of [RFC8342]), and <system> [I-D.ietf-netmod-system-config], if implemented.

2. The "ietf-restconf-client" Module

The RESTCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the RESTCONF client supports.

2.1. Data Model Overview

This section provides an overview of the "ietf-restconf-client" module in terms of its features and groupings.

2.1.1. Features

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

```
Features:
  +-- https-initiate
  +-- http-listen
  +-- https-listen
  +-- central-restconf-client-supported
```

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

2.1.2. Groupings

The "ietf-restconf-client" module defines the following "grouping" statements:

* restconf-client-initiate-stack-grouping
* restconf-client-listen-stack-grouping
* restconf-client-app-grouping

Each of these groupings are presented in the following subsections.

2.1.2.1. The "restconf-client-initiate-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-initiate-stack-grouping" grouping:

```
  grouping restconf-client-initiate-stack-grouping:
    +-- (transport)
       +--:(https) {https-initiate}?
          +-- https
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
             |  +---u httpc:http-client-grouping
             +-- restconf-client-parameters
                +---u rcc:restconf-client-grouping
```

Comments:

* The "restconf-client-initiate-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that initiate connections to RESTCONF servers, as opposed to receiving call-home [RFC8071] connections.

* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].

2.1.2.2. The "restconf-client-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-listen-stack-grouping" grouping:

```
  grouping restconf-client-listen-stack-grouping:
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-client-parameters
       |     |  +---u httpc:http-client-grouping
       |     +-- restconf-client-parameters
       |        +---u rcc:restconf-client-grouping
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-client-parameters
             |  +---u tlsc:tls-client-grouping
             +-- http-client-parameters
             |  +---u httpc:http-client-grouping
             +-- restconf-client-parameters
                +---u rcc:restconf-client-grouping
```

Comments:

* The "restconf-client-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that receive call-home [RFC8071] connections from RESTCONF servers.

* The "transport" choice node enables either the HTTP or HTTPS transports to be configured, with each option enabled by a "feature" statement. Note that RESTCONF requires HTTPS, the HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-client.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].

2.1.2.3. The "restconf-client-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-app-grouping" grouping:

```
grouping restconf-client-app-grouping:
  +-- initiate! {https-initiate}?
  |  +-- restconf-server* [name]
  |     +-- name?                 string
  |     +-- endpoints
  |     |  +-- endpoint* [name]
  |     |     +-- name?                                          string
  |     |     +---u restconf-client-initiate-stack-grouping
  |     +-- connection-type
  |     |  +-- (connection-type)
  |     |     +--:(persistent-connection)
  |     |     |  +-- persistent!
  |     |     +--:(periodic-connection)
  |     |        +-- periodic!
  |     |           +-- period?         uint16
  |     |           +-- anchor-time?    yang:date-and-time
  |     |           +-- idle-timeout?   uint16
  |     +-- reconnect-strategy
  |        +-- start-with?     enumeration
  |        +-- max-wait?       uint16
  |        +-- max-attempts?   uint8
  +-- listen! {http-listen or https-listen}?
     +-- idle-timeout?   uint16
     +-- endpoints
        +-- endpoint* [name]
           +-- name?                                       string
           +---u restconf-client-listen-stack-grouping
```

Comments:

* The "restconf-client-app-grouping" defines the configuration for a RESTCONF client that supports both initiating connections to RESTCONF servers as well as receiving call-home connections from RESTCONF servers.

* Both the "initiate" and "listen" subtrees are predicated by "feature" statements.

* For the referenced grouping statement(s):

  - The "restconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.1 in this document.
  - The "restconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.

2.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-restconf-client" module:

```
module: ietf-restconf-client
  +--rw restconf-client {central-restconf-client-supported}?
     +---u restconf-client-app-grouping
```

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 top-level node "restconf-client" is additionally constrained by the feature "central-restconf-client-supported".

* The "restconf-client-app-grouping" grouping is discussed in Section 2.1.2.3 in this document.

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

2.2. Example Usage

The following example illustrates configuring a RESTCONF client to initiate connections, as well as to listen for call-home connections.

This example is consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

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

<restconf-client xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-cl\
ient">

  <!-- RESTCONF servers to initiate connections to -->
  <initiate>
    <restconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <https>
```

```
              <tcp-client-parameters>
                <remote-address>corp-fw1.example.com</remote-address>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <central-keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </central-keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <central-truststore-reference>trusted-server-ca-ce\
rts</central-truststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <central-truststore-reference>trusted-server-ee-ce\
rts</central-truststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <client-identity>
                  <basic>
                    <user-id>bob</user-id>
                    <cleartext-password>example-secret</cleartext-pass\
word>
                  </basic>
                </client-identity>
              </http-client-parameters>
            </https>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
```

```
            <https>
              <tcp-client-parameters>
                <remote-address>corp-fw2.example.com</remote-address>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-client-parameters>
                <client-identity>
                  <certificate>
                    <central-keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </central-keystore-reference>
                  </certificate>
                </client-identity>
                <server-authentication>
                  <ca-certs>
                    <central-truststore-reference>trusted-server-ca-ce\
rts</central-truststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <central-truststore-reference>trusted-server-ee-ce\
rts</central-truststore-reference>
                  </ee-certs>
                </server-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-client-parameters>
              <http-client-parameters>
                <client-identity>
                  <basic>
                    <user-id>bob</user-id>
                    <cleartext-password>example-secret</cleartext-pass\
word>
                  </basic>
                </client-identity>
              </http-client-parameters>
            </https>
          </endpoint>
        </endpoints>
```

```
        <connection-type>
          <persistent/>
        </connection-type>
      </restconf-server>
    </initiate>

    <!-- endpoints to listen for RESTCONF Call Home connections on -->
    <listen>
      <endpoints>
        <endpoint>
          <name>Intranet-facing listener</name>
          <https>
            <tcp-server-parameters>
              <local-address>192.0.2.2</local-address>
            </tcp-server-parameters>
            <tls-client-parameters>
              <client-identity>
                <certificate>
                  <central-keystore-reference>
                    <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                    <certificate>ex-rsa-cert</certificate>
                  </central-keystore-reference>
                </certificate>
              </client-identity>
              <server-authentication>
                <ca-certs>
                  <central-truststore-reference>trusted-server-ca-cert\
  s</central-truststore-reference>
                </ca-certs>
                <ee-certs>
                  <central-truststore-reference>trusted-server-ee-cert\
  s</central-truststore-reference>
                </ee-certs>
              </server-authentication>
              <keepalives>
                <peer-allowed-to-send/>
              </keepalives>
            </tls-client-parameters>
            <http-client-parameters>
              <client-identity>
                <basic>
                  <user-id>bob</user-id>
                  <cleartext-password>example-secret</cleartext-passwo\
  rd>
                </basic>
              </client-identity>
            </http-client-parameters>
          </https>
```

```
        </endpoint>
      </endpoints>
    </listen>
  </restconf-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC8040], and [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-tls-client-server], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-client@2024-03-16.yang"

```
module ietf-restconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-client";
  prefix rcc;

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

  import ietf-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-http-client {
    prefix httpc;
    reference
      "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
  }
```

```
     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 <mailto:kent+ietf@watsen.net>";

     description
       "This module contains a collection of YANG definitions
        for configuring RESTCONF clients.

        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 IIII
        (https://www.rfc-editor.org/info/rfcIIII); 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 2024-03-16 {
       description
         "Initial version";
       reference
         "RFC IIII: RESTCONF Client and Server Models";
     }

     // Features

     feature https-initiate {
       description
         "The 'https-initiate' feature indicates that the RESTCONF
          client supports initiating HTTPS connections to RESTCONF
          servers. This feature exists as HTTPS might not be a
          mandatory to implement transport in the future.";
```

```
      reference
        "RFC 8040: RESTCONF Protocol";
    }

    feature http-listen {
      description
        "The 'http-listen' feature indicates that the RESTCONF client
         supports opening a port to listen for incoming RESTCONF
         server call-home connections using HTTP.  This feature
         exists as not all RESTCONF clients may support RESTCONF
         call home.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature https-listen {
      description
        "The 'https-listen' feature indicates that the RESTCONF client
         supports opening a port to listen for incoming RESTCONF
         server call-home connections using HTTPS.  This feature
         exists as not all RESTCONF clients may support RESTCONF
         call home.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature central-restconf-client-supported {
     description
        "The 'central-restconf-client-supported' feature indicates
         that the server that implements this module supports
         the top-level 'restconf-client' node.

         This feature is needed as some servers may want to use
         features defined in this module, which requires this
         module to be implemented, without having to support
         the top-level 'restconf-client' node.";
    }

    // Groupings

    grouping restconf-client-grouping {
      description
        "A reusable grouping for configuring a RESTCONF client
         without any consideration for how underlying transport
         sessions are established.

         This grouping currently does not define any nodes. It
         exists only so the model can be consistent with other
```

```
          'client-server' models.";
      }

      grouping restconf-client-initiate-stack-grouping {
        description
          "A reusable grouping for configuring a RESTCONF client
           'initiate' protocol stack for a single outbound connection.";

        choice transport {
          mandatory true;
          description
            "Selects between available transports.";
          case https {
            if-feature "https-initiate";
            container https {
              must 'tls-client-parameters/client-identity
                    or http-client-parameters/client-identity';
              description
                "TCP, TLS, HTTP, and RESTCONF configuration to
                 initiate a RESTCONF over HTTPS connection.";
              container tcp-client-parameters {
                description
                  "TCP-level client parameters to initiate
                   a RESTCONF over HTTPS connection.";
                uses tcpc:tcp-client-grouping {
                  refine "remote-port" {
                    default "443";
                    description
                      "The RESTCONF client will attempt to
                       connect to the IANA-assigned well-known
                       port value for 'https' (443) if no value
                       is specified.";
                  }
                }
              }
              container tls-client-parameters {
                description
                  "TLS-level client parameters to initiate
                   a RESTCONF over HTTPS connection.";
                uses tlsc:tls-client-grouping;
              }
              container http-client-parameters {
                description
                  "HTTP-level client parameters to initiate
                   a RESTCONF over HTTPS connection.";
                uses httpc:http-client-grouping;
              }
              container restconf-client-parameters {
```

```
            description
              "RESTCONF-level client parameters to initiate
               a RESTCONF over HTTPS connection.";
            uses rcc:restconf-client-grouping;
          }
        }
      }
    }
  } // restconf-client-initiate-stack-grouping

  grouping restconf-client-listen-stack-grouping {
    description
      "A reusable grouping for configuring a RESTCONF client
       'listen' protocol stack for listening on a single port.  The
       'listen' stack supports call home connections, as
       described in RFC 8071";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    choice transport {
      mandatory true;
      description
        "Selects between available transports.";
      case http {
        if-feature "http-listen";
        container http {
          description
            "TCP, HTTP, and RESTCONF configuration to
             listen for RESTCONF over HTTPS connections.

             This transport option is made available to support
             deployments where the TLS connections are terminated
             by another system (e.g., a load balancer) fronting
             the client.";
          container tcp-server-parameters {
            description
              "TCP-level server parameters to listen for
               RESTCONF over HTTP connections.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "4336";
                description
                  "The RESTCONF client will listen on the IANA-
                   assigned well-known port for 'restconf-ch-tls'
                   (4336) if no value is specified.";
              }
            }
          }
          container http-client-parameters {
```

```
            description
              "HTTP-level client parameters to listen for
               RESTCONF over HTTP connections.";
            uses httpc:http-client-grouping;
          }
          container restconf-client-parameters {
            description
              "RESTCONF-level client parameters to listen
               for RESTCONF over HTTP connections.";
            uses rcc:restconf-client-grouping;
          }
        }
      }
      case https {
        if-feature "https-listen";
        container https {
          must 'tls-client-parameters/client-identity
                or http-client-parameters/client-identity';
          description
            "TCP, TLS, HTTP, and RESTCONF configuration to
             listen for RESTCONF over HTTPS connections.";
          container tcp-server-parameters {
            description
              "TCP-level server parameters to listen
               for RESTCONF over HTTPS connections.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "4336";
                description
                  "The RESTCONF client will listen on the IANA-
                   assigned well-known port for 'restconf-ch-tls'
                   (4336) if no value is specified.";
              }
            }
          }
          container tls-client-parameters {
            description
              "TLS-level client parameters to listen
               for RESTCONF over HTTPS connections.";
            uses tlsc:tls-client-grouping;
          }
          container http-client-parameters {
            description
              "HTTP-level client parameters to listen
               for RESTCONF over HTTPS connections.";
            uses httpc:http-client-grouping;
          }
          container restconf-client-parameters {
```

```
            description
              "RESTCONF-level client parameters to listen
               for RESTCONF over HTTPS connections.";
            uses rcc:restconf-client-grouping;
          }
        }
      }
    }
  } // restconf-client-listen-stack-grouping

  grouping restconf-client-app-grouping {
    description
      "A reusable grouping for configuring a RESTCONF client
       application that supports both 'initiate' and 'listen'
       protocol stacks for a multiplicity of connections.";
    container initiate {
      if-feature "https-initiate";
      presence
        "Indicates that client-initiated connections have been
         configured.  This statement is present so the mandatory
         descendant nodes do not imply that this node must be
         configured.";
      description
        "Configures client initiating underlying TCP connections.";
      list restconf-server {
        key "name";
        min-elements 1;
        description
          "List of RESTCONF servers the RESTCONF client is to
           maintain simultaneous connections with.";
        leaf name {
          type string;
          description
            "An arbitrary name for the RESTCONF server.";
        }
        container endpoints {
          description
            "Container for a list of endpoints.";
          list endpoint {
            key "name";
            min-elements 1;
            ordered-by user;
            description
              "A non-empty user-ordered list of endpoints for this
               RESTCONF client to try to connect to in sequence.
               Defining more than one enables high-availability.";
            leaf name {
              type string;
```

```
            description
              "An arbitrary name for this endpoint.";
          }
          uses restconf-client-initiate-stack-grouping;
        }
      }
      container connection-type {
        description
          "Indicates the RESTCONF client's preference for how
           the RESTCONF connection is maintained.";
        choice connection-type {
          mandatory true;
          description
            "Selects between available connection types.";
          case persistent-connection {
            container persistent {
              presence
                "Indicates that a persistent connection is to be
                 maintained.";
              description
                "Maintain a persistent connection to the
                 RESTCONF server. If the connection goes down,
                 immediately start trying to reconnect to the
                 RESTCONF server, using the reconnection strategy.

                 This connection type minimizes any RESTCONF server
                 to RESTCONF client data-transfer delay, albeit
                 at the expense of holding resources longer.";
            }
          }
          case periodic-connection {
            container periodic {
              presence
                "Indicates that a periodic connection is to be
                 maintained.";
              description
                "Periodically connect to the RESTCONF server.

                 This connection type decreases resource
                 utilization, albeit with increased delay
                 in RESTCONF server to RESTCONF client
                 interactions.

                 The RESTCONF client SHOULD gracefully close
                 the underlying TLS connection upon completing
                 planned activities.

                 Connections are established at the same start
```

```
          time regardless how long the previous connection
          stayed open.

          In the case that the previous connection is
          still active, establishing a new connection
          is NOT RECOMMENDED.";
        leaf period {
          type uint16;
          units "minutes";
          default "60";
          description
            "Duration of time between periodic
             connections.";
        }
        leaf anchor-time {
          type yang:date-and-time {
            // constrained to minute-level granularity
            pattern '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|[1-2]'
                  + '[0-9]|3[0-1])T(0[0-9]|1[0-9]|2[0-3]):['
                  + '0-5][0-9]:00(Z|[\+\-]((1[0-3]|0[0-9]):'
                  + '([0-5][0-9])|14:00))?';
          }
          description
            "Designates a timestamp before or after which a
             series of periodic connections are determined.
             The periodic connections occur at a whole
             multiple interval from the anchor time.

             If an 'anchor-time' is not provided, then the
             server may implicitly set it to the time when
             this configuraton is applied (e.g., on boot).

             For example, for an anchor time is 15 minutes
             past midnight and a period interval of 24 hours,
             then a periodic connection will occur 15 minutes
             past midnight everyday.";
        }
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "180"; // three minutes
          description
            "Specifies the maximum number of seconds
             that the underlying TCP session may remain
             idle. A TCP session will be dropped if it
             is idle for an interval longer than this
             number of seconds If set to zero, then the
             RESTCONF client will never drop a session
```

```
                    because it is idle.";
                }
              }
            } // periodic-connection
          } // connection-type
        } // connection-type
        container reconnect-strategy {
          description
            "The reconnection strategy directs how a RESTCONF
             client reconnects to a RESTCONF server, after
             discovering its connection to the server has
             dropped, even if due to a reboot.  The RESTCONF
             client starts with the specified endpoint and
             tries to connect to it max-attempts times before
             trying the next endpoint in the list (round
             robin).";
          leaf start-with {
            type enumeration {
              enum first-listed {
                description
                  "Indicates that reconnections should start
                   with the first endpoint listed.";
              }
              enum last-connected {
                description
                  "Indicates that reconnections should start
                   with the endpoint last connected to.  If
                   no previous connection has ever been
                   established, then the first endpoint
                   configured is used.   RESTCONF clients
                   SHOULD be able to remember the last
                   endpoint connected to across reboots.";
              }
              enum random-selection {
                description
                  "Indicates that reconnections should start with
                   a random endpoint.";
              }
            }
            default "first-listed";
            description
              "Specifies which of the RESTCONF server's
               endpoints the RESTCONF client should start
               with when trying to connect to the RESTCONF
               server.";
          }
          leaf max-wait {
            type uint16 {
```

```
                range "1..max";
              }
              units "seconds";
              default "5";
              description
                "Specifies the amount of time in seconds after which,
                 if the connection is not established, an endpoint
                 connection attempt is considered unsuccessful.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default "3";
              description
                "Specifies the number times the RESTCONF client
                 tries to connect to a specific endpoint before
                 moving on to the next endpoint in the list
                 (round robin).";
            }
          }
        }
      } // initiate

      container listen {
        if-feature "http-listen or https-listen";
        presence
          "Indicates that client-listening ports have been configured.
           This statement is present so the mandatory descendant nodes
           do not imply that this node must be configured.";
        description
          "Configures the client to accept call-home TCP connections.";
        leaf idle-timeout {
          type uint16;
          units "seconds";
          default "180"; // three minutes
          description
            "Specifies the maximum number of seconds that an
             underlying TCP session may remain idle. A TCP session
             will be dropped if it is idle for an interval longer
             then this number of seconds.  If set to zero, then
             the server will never drop a session because it is
             idle.";
        }
        container endpoints {
          description
            "Container for a list of endpoints.";
          list endpoint {
```

```
            key "name";
            min-elements 1;
            description
              "List of endpoints to listen for RESTCONF connections.";
            leaf name {
              type string;
              description
                "An arbitrary name for the RESTCONF listen endpoint.";
            }
            uses restconf-client-listen-stack-grouping;
          }
        }
      } // listen
    } // restconf-client-app-grouping

    // Protocol accessible node for servers that implement this module.
    container restconf-client {
      if-feature central-restconf-client-supported;
      uses restconf-client-app-grouping;
      description
        "Top-level container for RESTCONF client configuration.";
    }
  }

  <CODE ENDS>
```

3. The "ietf-restconf-server" Module

The RESTCONF server model presented in this section supports both listening for connections as well as initiating call-home connections.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the RESTCONF server supports.

3.1. Data Model Overview

This section provides an overview of the "ietf-restconf-server" module in terms of its features and groupings.

3.1.1. Features

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

```
Features:
  +-- http-listen
  +-- https-listen
  +-- https-call-home
  +-- central-restconf-server-supported
```

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

3.1.2. Groupings

The "ietf-restconf-server" module defines the following "grouping" statements:

* restconf-server-grouping
* restconf-server-listen-stack-grouping
* restconf-server-callhome-stack-grouping
* restconf-server-app-grouping

Each of these groupings are presented in the following subsections.

3.1.2.1. The "restconf-server-grouping" Grouping

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

```
  grouping restconf-server-grouping:
    +-- client-identity-mappings
       +---u x509c2n:cert-to-name
```

Comments:

* The "restconf-server-grouping" defines the configuration for the "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).

* The "client-identity-mappings" node defines a mapping from certificate fields to RESTCONF user names.

* For the referenced grouping statement(s):

  - The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

3.1.2.2. The "restconf-server-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-listen-stack-grouping" grouping:

```
  grouping restconf-server-listen-stack-grouping:
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- external-endpoint!
       |     |  +-- address    inet:host
       |     |  +-- port?      inet:port-number
       |     +-- tcp-server-parameters
       |     |  +---u tcps:tcp-server-grouping
       |     +-- http-server-parameters
       |     |  +---u https:http-server-grouping
       |     +-- restconf-server-parameters
       |        +---u rcs:restconf-server-grouping
       +--:(https) {https-listen}?
          +-- https
             +-- tcp-server-parameters
             |  +---u tcps:tcp-server-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
             |  +---u https:http-server-grouping
             +-- restconf-server-parameters
                +---u rcs:restconf-server-grouping
```

Comments:

* The "restconf-server-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack for RESTCONF servers that listen for connections from RESTCONF clients, as opposed to initiating call-home [RFC8071] connections.

* The "transport" choice node enables either the HTTP or HTTPS transports to be configured, with each option enabled by a "feature" statement. The HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-server.

* For the referenced grouping statement(s):

  - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].

- The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
- The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.3. The "restconf-server-callhome-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-callhome-stack-grouping" grouping:

```
  grouping restconf-server-callhome-stack-grouping:
    +-- (transport)
       +--:(https) {https-call-home}?
          +-- https
             +-- tcp-client-parameters
             |  +---u tcpc:tcp-client-grouping
             +-- tls-server-parameters
             |  +---u tlss:tls-server-grouping
             +-- http-server-parameters
             |  +---u https:http-server-grouping
             +-- restconf-server-parameters
                +---u rcs:restconf-server-grouping
```

Comments:

* The "restconf-server-callhome-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF servers that initiate call-home [RFC8071] connections to RESTCONF clients.

* The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.

* For the referenced grouping statement(s):

  - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
  - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tls-client-server].
  - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
  - The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.4. The "restconf-server-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-app-grouping" grouping:

```
  grouping restconf-server-app-grouping:
    +-- listen! {http-listen or https-listen}?
    |  +-- endpoints
    |     +-- endpoint* [name]
    |        +-- name?                                        string
    |        +---u restconf-server-listen-stack-grouping
    +-- call-home! {https-call-home}?
       +-- restconf-client* [name]
          +-- name?                 string
          +-- endpoints
          |  +-- endpoint* [name]
          |     +-- name?                                          string
          |     +---u restconf-server-callhome-stack-grouping
          +-- connection-type
          |  +-- (connection-type)
          |     +--:(persistent-connection)
          |     |  +-- persistent!
          |     +--:(periodic-connection)
          |        +-- periodic!
          |           +-- period?         uint16
          |           +-- anchor-time?    yang:date-and-time
          |           +-- idle-timeout?   uint16
          +-- reconnect-strategy
             +-- start-with?     enumeration
             +-- max-wait?       uint16
             +-- max-attempts?   uint8
```

Comments:

* The "restconf-server-app-grouping" defines the configuration for a RESTCONF server that supports both listening for connections from RESTCONF clients as well as initiating call-home connections to RESTCONF clients.

* Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.

* For the referenced grouping statement(s):

  - The "restconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
  - The "restconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-restconf-server" module:

```
module: ietf-restconf-server
  +--rw restconf-server {central-restconf-server-supported}?
     +---u restconf-server-app-grouping
```

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 top-level node "restconf-server" is additionally constrained by the feature "central-restconf-server-supported".

* The "restconf-server-app-grouping" grouping is discussed in Section 3.1.2.4 in this document.

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

3.2. Example Usage

The following example illustrates configuring a RESTCONF server to listen for RESTCONF client connections, as well as configuring call-home to one RESTCONF client.

This example is consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

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

<restconf-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-server"
  xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">

  <!-- endpoints to listen for RESTCONF connections on -->
  <listen>
    <endpoints>
      <endpoint>
        <name>restconf/https</name>
        <https>
```

```
            <tcp-server-parameters>
              <local-address>192.0.2.2</local-address>
            </tcp-server-parameters>
            <tls-server-parameters>
              <server-identity>
                <certificate>
                  <central-keystore-reference>
                    <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                    <certificate>ex-rsa-cert</certificate>
                  </central-keystore-reference>
                </certificate>
              </server-identity>
              <client-authentication>
                <ca-certs>
                  <central-truststore-reference>trusted-client-ca-cert\
  s</central-truststore-reference>
                </ca-certs>
                <ee-certs>
                  <central-truststore-reference>trusted-client-ee-cert\
  s</central-truststore-reference>
                </ee-certs>
              </client-authentication>
              <keepalives>
                <peer-allowed-to-send/>
              </keepalives>
            </tls-server-parameters>
            <http-server-parameters>
              <server-name>foo.example.com</server-name>
            </http-server-parameters>
            <restconf-server-parameters>
              <client-identity-mappings>
                <cert-to-name>
                  <id>1</id>
                  <fingerprint>11:0A:05:11:00</fingerprint>
                  <map-type>x509c2n:specified</map-type>
                  <name>scooby-doo</name>
                </cert-to-name>
                  <cert-to-name>
                  <id>2</id>
                  <map-type>x509c2n:san-any</map-type>
                </cert-to-name>
              </client-identity-mappings>
            </restconf-server-parameters>
          </https>
        </endpoint>
      </endpoints>
    </listen>
```

```
     <!-- call home to a RESTCONF client with two endpoints -->
     <call-home>
       <restconf-client>
         <name>config-manager</name>
         <endpoints>
           <endpoint>
             <name>east-data-center</name>
             <https>
               <tcp-client-parameters>
                 <remote-address>east.example.com</remote-address>
                 <keepalives>
                   <idle-time>7200</idle-time>
                   <max-probes>9</max-probes>
                   <probe-interval>75</probe-interval>
                 </keepalives>
               </tcp-client-parameters>
               <tls-server-parameters>
                 <server-identity>
                   <certificate>
                     <central-keystore-reference>
                       <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
   ey>
                       <certificate>ex-rsa-cert</certificate>
                     </central-keystore-reference>
                   </certificate>
                 </server-identity>
                 <client-authentication>
                   <ca-certs>
                     <central-truststore-reference>trusted-client-ca-ce\
   rts</central-truststore-reference>
                   </ca-certs>
                   <ee-certs>
                     <central-truststore-reference>trusted-client-ee-ce\
   rts</central-truststore-reference>
                   </ee-certs>
                 </client-authentication>
                 <keepalives>
                   <test-peer-aliveness>
                     <max-wait>30</max-wait>
                     <max-attempts>3</max-attempts>
                   </test-peer-aliveness>
                 </keepalives>
               </tls-server-parameters>
               <http-server-parameters>
                 <server-name>foo.example.com</server-name>
               </http-server-parameters>
               <restconf-server-parameters>
                 <client-identity-mappings>
```

```
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </restconf-server-parameters>
            </https>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <https>
              <tcp-client-parameters>
                <remote-address>west.example.com</remote-address>
                <keepalives>
                  <idle-time>7200</idle-time>
                  <max-probes>9</max-probes>
                  <probe-interval>75</probe-interval>
                </keepalives>
              </tcp-client-parameters>
              <tls-server-parameters>
                <server-identity>
                  <certificate>
                    <central-keystore-reference>
                      <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                      <certificate>ex-rsa-cert</certificate>
                    </central-keystore-reference>
                  </certificate>
                </server-identity>
                <client-authentication>
                  <ca-certs>
                    <central-truststore-reference>trusted-client-ca-ce\
rts</central-truststore-reference>
                  </ca-certs>
                  <ee-certs>
                    <central-truststore-reference>trusted-client-ee-ce\
rts</central-truststore-reference>
                  </ee-certs>
                </client-authentication>
                <keepalives>
                  <test-peer-aliveness>
                    <max-wait>30</max-wait>
```

```
                    <max-attempts>3</max-attempts>
                  </test-peer-aliveness>
                </keepalives>
              </tls-server-parameters>
              <http-server-parameters>
                <server-name>foo.example.com</server-name>
              </http-server-parameters>
              <restconf-server-parameters>
                <client-identity-mappings>
                  <cert-to-name>
                    <id>1</id>
                    <fingerprint>11:0A:05:11:00</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                  <cert-to-name>
                    <id>2</id>
                    <map-type>x509c2n:san-any</map-type>
                  </cert-to-name>
                </client-identity-mappings>
              </restconf-server-parameters>
            </https>
          </endpoint>
        </endpoints>
        <connection-type>
          <periodic>
            <idle-timeout>300</idle-timeout>
            <anchor-time>2023-03-15T01:30:00Z</anchor-time>
            <period>60</period>
          </periodic>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
          <max-wait>3</max-wait>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </restconf-client>
    </call-home>
  </restconf-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC7407], [RFC8040], [RFC8071], [I-D.ietf-netconf-tcp-client-server], [I-D.ietf-netconf-tls-client-server], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-server@2024-03-16.yang"

```
   module ietf-restconf-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-server";
     prefix rcs;

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

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

     import ietf-x509-cert-to-name {
       prefix x509c2n;
       reference
         "RFC 7407: A YANG Data Model for SNMP Configuration";
     }

     import ietf-tcp-client {
       prefix tcpc;
       reference
         "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
     }

     import ietf-tcp-server {
       prefix tcps;
       reference
         "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
     }

     import ietf-tls-server {
       prefix tlss;
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
     }

     import ietf-http-server {
       prefix https;
       reference
         "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
     }

     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 <mailto:kent+ietf@watsen.net>";

    description
      "This module contains a collection of YANG definitions
       for configuring RESTCONF servers.

       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 IIII
       (https://www.rfc-editor.org/info/rfcIIII); 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 2024-03-16 {
      description
        "Initial version";
      reference
        "RFC IIII: RESTCONF Client and Server Models";
    }

    // Features

    feature http-listen {
      description
        "The 'http-listen' feature indicates that the RESTCONF server
         supports opening a port to listen for incoming RESTCONF over
         TCP client connections, whereby the TLS connections are
         terminated by an external system.";
      reference
```

```
        "RFC 8040: RESTCONF Protocol";
    }

    feature https-listen {
      description
        "The 'https-listen' feature indicates that the RESTCONF server
         supports opening a port to listen for incoming RESTCONF over
         TLS client connections, whereby the TLS connections are
         terminated by the server itself.";
      reference
        "RFC 8040: RESTCONF Protocol";
    }

    feature https-call-home {
      description
        "The 'https-call-home' feature indicates that the RESTCONF
         server supports initiating connections to RESTCONF clients.";
      reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    feature central-restconf-server-supported {
     description
        "The 'central-restconf-server-supported' feature indicates
         that the server supports the top-level 'restconf-server'
         node.

         This feature is needed as some servers may want to use
         features defined in this module, which requires this
         module to be implemented, without having to support
         the top-level 'restconf-server' node.";
    }

    // Groupings

    grouping restconf-server-grouping {
      description
        "A reusable grouping for configuring a RESTCONF server
         without any consideration for how underlying transport
         sessions are established.

         Note that this grouping uses a fairly typical descendant
         node name such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue by wrapping the 'uses'
         statement in a container called, e.g.,
         'restconf-server-parameters'.  This model purposely does
         not do this itself so as to provide maximum flexibility
```

```
          to consuming models.";

        container client-identity-mappings {
          description
            "Specifies mappings through which RESTCONF client X.509
             certificates are used to determine a RESTCONF username.
             If no matching and valid cert-to-name list entry can be
             found, then the RESTCONF server MUST close the connection,
             and MUST NOT accept RESTCONF messages over it.";
          reference
            "RFC 7407: A YANG Data Model for SNMP Configuration.";
          uses x509c2n:cert-to-name {
            refine "cert-to-name/fingerprint" {
              mandatory false;
              description
                "A 'fingerprint' value does not need to be specified
                 when the 'cert-to-name' mapping is independent of
                 fingerprint matching.  A 'cert-to-name' having no
                 fingerprint value will match any client certificate
                 and therefore should only be present at the end of
                 the user-ordered 'cert-to-name' list.";
            }
          }
        }
      }

      grouping restconf-server-listen-stack-grouping {
        description
          "A reusable grouping for configuring a RESTCONF server
           'listen' protocol stack for listening on a single port.";
        choice transport {
          mandatory true;
          description
            "Selects between available transports.";
          case http {
            if-feature "http-listen";
            container http {
              description
                "Configures RESTCONF server stack assuming that
                 TLS-termination is handled externally.

                 How a RESTCONF-server identifies RESTCONF-clients
                 authenticating using a TLS-level client-certificate
                 with external TLS termination is out of scope of
                 this document.";
              container external-endpoint {
                presence
                  "Identifies that an external endpoint has been
```

```
            configured.  This statement is present so the
            mandatory descendant nodes do not imply that
            this node must be configured.";
          description
            "Identifies contact information for the external
            system that terminates connections before passing
            them through to this server (e.g., a network address
            translator or a load balancer).  These values have
            no effect on the local operation of this server,
            but may be used by the application when needing to
            inform other systems how to contact this server.";
          leaf address {
            type inet:host;
            mandatory true;
            description
              "The IP address or hostname of the external
               system that terminates incoming RESTCONF
               client connections before forwarding them
               to this server.";
          }
          leaf port {
            type inet:port-number;
            default "443";
            description
              "The port number that the external system listens
               on for incoming RESTCONF client connections that
               are forwarded to this server.  The default HTTPS
               port (443) is used, as expected for a RESTCONF
               connection.";
          }
        }
        container tcp-server-parameters {
          description
            "TCP-level server parameters to listen for
             RESTCONF over HTTP connections.";
          uses tcps:tcp-server-grouping {
            refine "local-port" {
              default "80";
              description
                "The RESTCONF server will listen on the IANA-
                 assigned well-known port value for 'http'
                 (80) if no value is specified.";
            }
          }
        }
        container http-server-parameters {
          description
            "HTTP-level server parameters to listen
```

```
                for RESTCONF over HTTP connections.";
              uses https:http-server-grouping;
            }
            container restconf-server-parameters {
              description
                "RESTCONF-level server parameters to listen
                 for RESTCONF over HTTP connections.";
              uses rcs:restconf-server-grouping;
            }
          }
        }
        case https {
          if-feature "https-listen";
          container https {
            description
              "Configures RESTCONF server stack assuming that
               TLS-termination is handled internally (i.e.,
               not by a TLS-terminator in front of the RESTCONF
               server).";
            container tcp-server-parameters {
              description
                "TCP-level server parameters to listen for
                 RESTCONF over HTTPS connections.";
              uses tcps:tcp-server-grouping {
                refine "local-port" {
                  default "443";
                  description
                    "The RESTCONF server will listen on the IANA-
                     assigned well-known port value for 'https'
                     (443) if no value is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "TLS-level server parameters to listen
                 for RESTCONF over HTTPS connections.";
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "HTTP-level server parameters to listen
                 for RESTCONF over HTTPS connections.";
              uses https:http-server-grouping;
            }
            container restconf-server-parameters {
              description
                "RESTCONF-level server parameters to listen
```

```
                for RESTCONF over HTTPS connections.";
              uses rcs:restconf-server-grouping;
            }
          }
        }
      }
    }

    grouping restconf-server-callhome-stack-grouping {
      description
        "A reusable grouping for configuring a RESTCONF server
         'call-home' protocol stack, for a single outbound
         connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case https {
          if-feature "https-call-home";
          container https {
            description
              "Configures RESTCONF server stack assuming that
               TLS-termination is handled internally.";
            container tcp-client-parameters {
              description
                "TCP-level client parameters to initiate a
                 RESTCONF over HTTPS Call Home connection.";
              uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                  default "4336";
                  description
                    "The RESTCONF server will attempt to
                     connect to the IANA-assigned well-known
                     port for 'restconf-ch-tls' (4336) if no
                     value is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "TLS-level server parameters to initiate a
                 RESTCONF over HTTPS Call Home connection.";
              uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "HTTP-level server parameters to initiate a
                 RESTCONF over HTTPS Call Home connection.";
```

```
            uses https:http-server-grouping;
          }
          container restconf-server-parameters {
            description
              "RESTCONF-level server parameters to initiate
               a RESTCONF over HTTPS Call Home connection.";
            uses rcs:restconf-server-grouping;
          }
        }
      }
    }
  }

  grouping restconf-server-app-grouping {
    description
      "A reusable grouping for configuring a RESTCONF server
       application that supports both 'listen' and 'call-home'
       protocol stacks for a multiplicity of connections.";
    container listen {
      if-feature "http-listen or https-listen";
      presence
        "Identifies that the server has been configured to
         listen for incoming client connections.  This statement
         is present so the mandatory descendant nodes do not
         imply that this node must be configured.";
      description
        "Configures the RESTCONF server to listen for RESTCONF
         client connections.";
      container endpoints {
        description
          "Container for a list of endpoints.";
        list endpoint {
          key "name";
          min-elements 1;
          description
            "List of endpoints to listen for RESTCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF listen endpoint.";
          }
          uses restconf-server-listen-stack-grouping;
        }
      }
    }
    container call-home {
      if-feature "https-call-home";
      presence
```

```
          "Identifies that the server has been configured to initiate
           call home connections.  This statement is present so the
           mandatory descendant nodes do not imply that this node
           must be configured.";
        description
          "Configures the RESTCONF server to initiate the underlying
           transport connection to RESTCONF clients.";
        list restconf-client {
          key "name";
          min-elements 1;
          description
            "List of RESTCONF clients the RESTCONF server is to
             maintain simultaneous call-home connections with.";
          leaf name {
            type string;
            description
              "An arbitrary name for the remote RESTCONF client.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key "name";
              min-elements 1;
              ordered-by user;
              description
                "User-ordered list of endpoints for this RESTCONF
                 client.  Defining more than one enables high-
                 availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this endpoint.";
              }
              uses restconf-server-callhome-stack-grouping;
            }
          }
          container connection-type {
            description
              "Indicates the RESTCONF server's preference for how the
               RESTCONF connection is maintained.";
            choice connection-type {
              mandatory true;
              description
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
                  presence
```

```
                    "Indicates that a persistent connection is to be
                     maintained.";
                  description
                    "Maintain a persistent connection to the RESTCONF
                     client. If the connection goes down, immediately
                     start trying to reconnect to the RESTCONF client,
                     using the reconnection strategy.

                     This connection type minimizes any RESTCONF
                     client to RESTCONF server data-transfer delay,
                     albeit at the expense of holding resources
                     longer.";
                }
              }
              case periodic-connection {
                container periodic {
                  presence
                    "Indicates that a periodic connection is to be
                     maintained.";
                  description
                    "Periodically connect to the RESTCONF client.

                     This connection type decreases resource
                     utilization, albeit with increased delay in
                     RESTCONF client to RESTCONF server interactions.

                     The RESTCONF client SHOULD gracefully close
                     the underlying TLS connection upon completing
                     planned activities.  If the underlying TLS
                     connection is not closed gracefully, the
                     RESTCONF server MUST immediately attempt
                     to reestablish the connection.

                     Connections are established at the same start
                     time regardless how long the previous connection
                     stayed open.

                     In the case that the previous connection is
                     still active (i.e., the RESTCONF client has not
                     closed it yet), establishing a new connection
                     is NOT RECOMMENDED.";

                  leaf period {
                    type uint16;
                    units "minutes";
                    default "60";
                    description
                      "Duration of time between periodic connections.";
```

```
                  }
                  leaf anchor-time {
                    type yang:date-and-time {
                      // constrained to minute-level granularity
                      pattern '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|[1-2]'
                            + '[0-9]|3[0-1])T(0[0-9]|1[0-9]|2[0-3]):['
                            + '0-5][0-9]:00(Z|[\+\-]((1[0-3]|0[0-9]):'
                            + '([0-5][0-9])|14:00))?';
                    }
                    description
                      "Designates a timestamp before or after which a
                       series of periodic connections are determined.
                       The periodic connections occur at a whole
                       multiple interval from the anchor time.

                       If an 'anchor-time' is not provided, then the
                       server may implicitly set it to the time when
                       this configuraton is applied (e.g., on boot).

                       For example, for an anchor time is 15 minutes
                       past midnight and a period interval of 24 hours,
                       then a periodic connection will occur 15 minutes
                       past midnight everyday.";
                  }
                  leaf idle-timeout {
                    type uint16;
                    units "seconds";
                    default "180"; // three minutes
                    description
                      "Specifies the maximum number of seconds that
                       the underlying TCP session may remain idle.
                       A TCP session will be dropped if it is idle
                       for an interval longer than this number of
                       seconds.  If set to zero, then the server
                       will never drop a session because it is idle.";
                  }
                }
              }
            }
          }
          container reconnect-strategy {
            description
              "The reconnection strategy directs how a RESTCONF server
               reconnects to a RESTCONF client after discovering its
               connection to the client has dropped, even if due to a
               reboot.  The RESTCONF server starts with the specified
               endpoint and tries to connect to it max-attempts times
               before trying the next endpoint in the list (round
```

```
           robin).";
        leaf start-with {
          type enumeration {
            enum first-listed {
              description
                "Indicates that reconnections should start with
                 the first endpoint listed.";
            }
            enum last-connected {
              description
                "Indicates that reconnections should start with
                 the endpoint last connected to.  If no previous
                 connection has ever been established, then the
                 first endpoint configured is used.   RESTCONF
                 servers SHOULD be able to remember the last
                 endpoint connected to across reboots.";
            }
            enum random-selection {
              description
                "Indicates that reconnections should start with
                 a random endpoint.";
            }
          }
          default "first-listed";
          description
            "Specifies which of the RESTCONF client's endpoints
             the RESTCONF server should start with when trying
             to connect to the RESTCONF client.";
        }
        leaf max-wait {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default "5";
          description
            "Specifies the amount of time in seconds after which,
             if the connection is not established, an endpoint
             connection attempt is considered unsuccessful.";
        }
        leaf max-attempts {
          type uint8 {
            range "1..max";
          }
          default "3";
          description
            "Specifies the number times the RESTCONF server tries
             to connect to a specific endpoint before moving on to
```

```
                    the next endpoint in the list (round robin).";
                }
              }
            } // restconf-client
          } // call-home
        } // restconf-server-app-grouping

        // Protocol accessible node for servers that implement this module.
        container restconf-server {
          if-feature central-restconf-server-supported;
          uses restconf-server-app-grouping;
          description
            "Top-level container for RESTCONF server configuration.";
        }
      }
```

<CODE ENDS>

4. Security Considerations

4.1. Considerations for the "ietf-restconf-client" YANG Module

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

The "ietf-restconf-client" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

4.2. Considerations for the "ietf-restconf-server" YANG Module

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

The "ietf-restconf-server" YANG module defines data nodes that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-restconf-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-client
prefix:       rcc
reference:    RFC IIII

name:         ietf-restconf-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-server
prefix:       rcs
reference:    RFC IIII
```

6. References

6.1. Normative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

6.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-

ietf-netmod-system-config-05, 21 February 2024,
<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

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

[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>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

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

[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>.

[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>.

Appendix A. Change Log

A.1. 00 to 01

* Renamed "keychain" to "keystore".

A.2. 01 to 02

* Filled in previously missing 'ietf-restconf-client' module.

* Updated the ietf-restconf-server module to accommodate new grouping 'ietf-tls-server-grouping'.

A.3. 02 to 03

* Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.

* Changed restconf-client??? to be a grouping (not a container).

A.4. 03 to 04

* Added RFC 8174 to Requirements Language Section.

* Replaced refine statement in ietf-restconf-client to add a mandatory true.

* Added refine statement in ietf-restconf-server to add a must statement.

* Now there are containers and groupings, for both the client and server models.

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated examples to inline key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated examples to inline key and certificates (no longer a leafref to keystore)

A.6. 05 to 06

* Fixed change log missing section issue.

* Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.

* Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

* removed "idle-timeout" from "persistent" connection config.

* Added "random-selection" for reconnection-strategy's "starts-with" enum.

* Replaced "connection-type" choice default (persistent) with "mandatory true".

* Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.

* Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

* Modified examples to be compatible with new crypto-types algs

A.9. 08 to 09

* Corrected use of "mandatory true" for "address" leafs.

* Updated examples to reflect update to groupings defined in the keystore draft.

* Updated to use groupings defined in new TCP and HTTP drafts.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

* Reformatted YANG modules.

A.11. 10 to 11

* Adjusted for the top-level "demux container" added to groupings imported from other modules.

* Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Moved "expanded" tree diagrams to the Appendix.

A.12. 11 to 12

* Removed the 'must' statement limiting keepalives in periodic connections.

* Updated models and examples to reflect removal of the "demux" containers in the imported models.

* Updated the "periodic-connnection" description statements to better describe behavior when connections are not closed gracefully.

* Updated text to better reference where certain examples come from (e.g., which Section in which draft).

* In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.

* Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

A.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

* In ietf-restconf-server, Added 'http-listen' (not https-listen) choice, to support case when server is behind a TLS-terminator.

* Refactored server module to be more like other 'server' models. If folks like it, will also apply to the client model, as well as to both the netconf client/server models. Now the 'restconf-server-grouping' is just the RC-specific bits (i.e., the "demux" container minus the container), 'restconf-server-[listen|callhome]-stack-grouping' is the protocol stack for a single connection, and 'restconf-server-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

A.14. 13 to 14

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

* Adjusting from change in TLS client model (removing the top-level 'certificate' container).

* Added "external-endpoint" to the "http-listen" choice in ietf-restconf-server.

A.15. 14 to 15

* Added missing "or https-listen" clause in a "must" expression.

* Refactored the client module similar to how the server module was refactored in -13. Now the 'restconf-client-grouping' is just the RC-specific bits, the 'restconf-client-[initiate|listen]-stack-grouping' is the protocol stack for a single connection, and 'restconf-client-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

A.16. 15 to 16

* Added refinement to make "cert-to-name/fingerprint" be mandatory false.

* Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

* Updated restconf-client example to reflect that http-client-grouping no longer has a "protocol-version" leaf.

A.17. 16 to 17

* Updated examples to include the "*-key-format" nodes.

* Updated examples to remove the "required" nodes.

A.18. 17 to 18

* Updated examples to reflect new "bag" addition to truststore.

A.19. 18 to 19

* Updated examples to remove the 'algorithm' nodes.

* Updated examples to reflect the new TLS keepalives structure.

* Removed the 'protocol-versions' node from the restconf-server examples.

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

A.20. 19 to 20

* Moved and changed "must" statement so that either TLS *or* HTTP auth must be configured.

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

* Updated the Security Considerations section.

A.21. 20 to 21

* Cleaned up titles in the IANA Consideratons section
* Fixed issues found by the SecDir review of the "keystore" draft.

A.22. 21 to 22

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.23. 22 to 23

* Further clarified why some 'presence' statements are present.
* Addressed nits found in YANG Doctor reviews.
* Aligned modules with `pyang -f` formatting.

A.24. 23 to 24

* Removed Appendix A with fully-expanded tree diagrams.
* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
* Minor editorial nits

A.25. 24 to 25

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)
* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

A.26. 25 to 26

* Added feature "central-restconf-client-supported" to top-level node "restconf-client".
* Added feature "central-restconf-server-supported" to top-level node "restconf-server".

A.27. 26 to 27

* Updated per Shepherd reviews impacting the suite of drafts.
* Added "max-wait" leaf to the "reconnect-strategy" nodes.

A.28. 27 to 28

* Updated per Shepherd reviews impacting the suite of drafts.

A.29. 28 to 29

* Updated (implicitly) via Tom Petch reviews.

* Fixed pattern statement for "leaf anchor-time".

* Updated examples to use IETF-sanctioned values.

A.30. 29 to 30

* Addresses AD review comments.

* Added note to Editor to fix line foldings.

* Removed "Conventions" section as there are no "BASE64VALUE=" values used in draft.

* Removed restconf-client-grouping, since it was empty.

* Removed erronious statement "client-identity-mappings" must be enabled by a "feature".

* Added Security Considerations text to also look a SC-section from imported modules.

* Removed "A wrapper around the foobar parameters to avoid name collisions" text.

* Added container "endpoints" to wrap list "endpoint".

* Fixed if-feature "https-listen" to if-feature "https-call-home".

A.31. 30 to 31

* Addresses AD review by Rob Wilton.

A.32. 31 to 32

* Addresses 1st-round of IESG reviews.

A.33. 32 to 34

* Addresses issues found in OpsDir review of the ssh-client-server draft.

* s/defines/presents/ in a few places.

* Add refs to where the 'operational' and 'system' datastores are defined.

* Renamed Security Considerations section s/Template for/ Considerations for/

A.34. 34 to 36

* Nothing changed. Only bumped for automation...

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balázs Kovács, Benoit Claise, Bert Wijnen David Lamparter, Jürgen Schönwälder, Ladislav Lhotka, Martin Björklund, Qiufang Ma, Mehmet Ersue, Michal Vako, Phil Shafer, Qiufang Ma, Radek Krejci, Ramkumar Dhanapal, Rob Wilton, Sean Turner, and Tom Petch.

Author's Address

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

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

K. Watsen
Watsen Networks
G. Wu
Cisco Systems

10 July 2020

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-21

Abstract

This document defines three YANG modules: the first defines groupings for a generic SSH client, the second defines groupings for a generic SSH server, and the third defines common identities and groupings used by both the client and the server. It is intended that these groupings will be used by applications using the SSH protocol.

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 draft-ietf-netconf-trust-anchors

* "CCCC" --> the assigned RFC value for draft-ietf-netconf-keystore

* "DDDD" --> the assigned RFC value for draft-ietf-netconf-tcp-client-server

* "EEEE" --> the assigned RFC value for this draft

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

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

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

* Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines a grouping for a generic SSH client, the second defines a grouping for a generic SSH server, and the third defines identities and groupings common to both the client and the server. It is intended that these groupings will be used by applications using the SSH protocol [RFC4252], [RFC4253], and [RFC4254]. For instance, these groupings could be used to help define the data model for an OpenSSH [OPENSSH] server or a NETCONF over SSH [RFC6242] based server.

The client and server YANG modules in this document each define one grouping, which is focused on just SSH-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen on or connect to. This affords applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "ssh-server-grouping" grouping for the SSH parts it provides, while adding data nodes for the TCP-level call-home configuration.

The modules defined in this document use groupings defined in [I-D.ietf-netconf-keystore] enabling keys to be either locally defined or a reference to globally configured values.

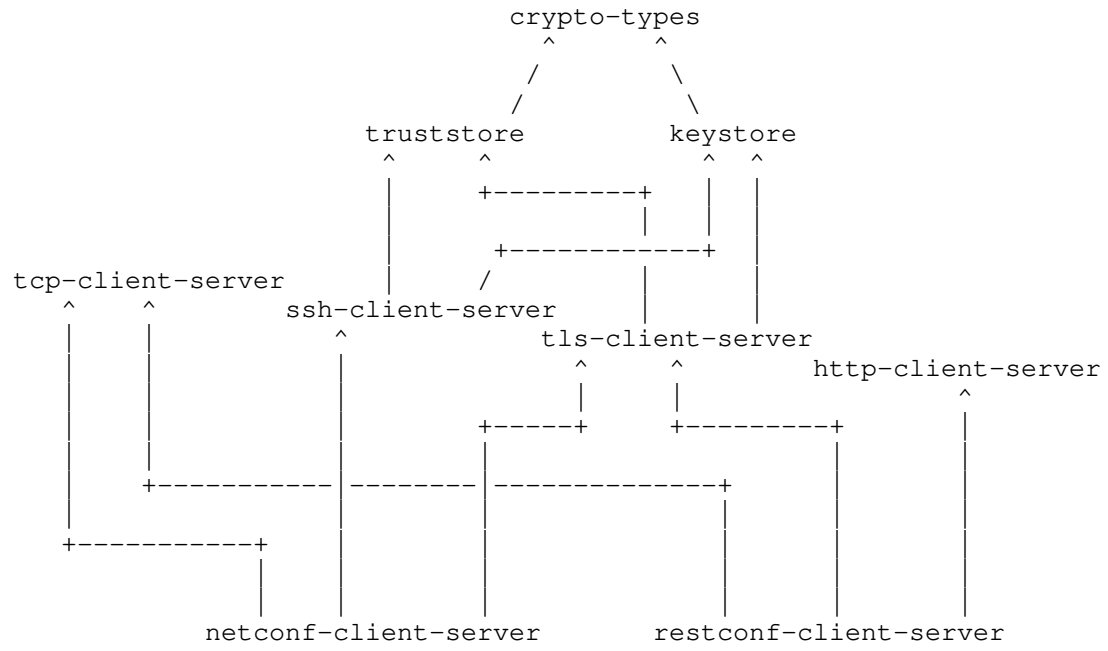
The modules defined in this document optionally support [RFC6187] enabling X.509v3 certificate based host keys and public keys.

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

2. The "ietf-ssh-common" Module

The SSH common model presented in this section contains identities and groupings common to both SSH clients and SSH servers. The "transport-params-grouping" grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the SSH transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive. As well, some algorithms that are REQUIRED by [RFC4253] are missing, notably "ssh-dss" and "diffie-hellman-group1-sha1" due to their weak security and there being alternatives that are widely supported.

2.1. Data Model Overview

2.1.1. Features

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

```
   Features:
     +-- ssh-ecc
     +-- ssh-x509-certs
     +-- ssh-dh-group-exchange
     +-- ssh-ctr
     +-- ssh-sha2
```

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-ssh-common" module:

```
   Identities:
     +-- public-key-alg-base
     |  +-- ssh-dss
     |  +-- ssh-rsa
     |  +-- ecdsa-sha2-nistp256
     |  +-- ecdsa-sha2-nistp384
     |  +-- ecdsa-sha2-nistp521
     |  +-- x509v3-ssh-rsa
     |  +-- x509v3-rsa2048-sha256
     |  +-- x509v3-ecdsa-sha2-nistp256
     |  +-- x509v3-ecdsa-sha2-nistp384
     |  +-- x509v3-ecdsa-sha2-nistp521
     +-- key-exchange-alg-base
     |  +-- diffie-hellman-group14-sha1
     |  +-- diffie-hellman-group-exchange-sha1
     |  +-- diffie-hellman-group-exchange-sha256
     |  +-- ecdh-sha2-nistp256
     |  +-- ecdh-sha2-nistp384
     |  +-- ecdh-sha2-nistp521
     +-- encryption-alg-base
     |  +-- triple-des-cbc
     |  +-- aes128-cbc
     |  +-- aes192-cbc
     |  +-- aes256-cbc
     |  +-- aes128-ctr
     |  +-- aes192-ctr
     |  +-- aes256-ctr
     +-- mac-alg-base
        +-- hmac-sha1
        +-- hmac-sha2-256
        +-- hmac-sha2-512
```

Comments:

* The diagram shows that there are four base identities.

* These identities are used by this module to define algorithms for public-key, key-exchange, encryption, and MACs.
* These base identities are "abstract", in the object orientied programming sense, in that they only define a "class" of algorithms, rather than a specific algorithm.

2.1.3. Groupings

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

```
Groupings:
  +-- transport-params-grouping
```

Each of these groupings are presented in the following subsections.

2.1.3.1. The "transport-params-grouping" Grouping

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

```
  grouping transport-params-grouping
    +-- host-key
    |  +-- host-key-alg*   identityref
    +-- key-exchange
    |  +-- key-exchange-alg*   identityref
    +-- encryption
    |  +-- encryption-alg*   identityref
    +-- mac
       +-- mac-alg*   identityref
```

Comments:

* This grouping is used by both the "ssh-client-grouping" and the "ssh-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.

* This grouping enables client and server configurations to specify the algorithms that are to be used when establishing SSH sessions.

* Each list is "ordered-by user".

2.1.4. Protocol-accessible Nodes

The "ietf-ssh-common" module does not contain any protocol-accessible nodes, but the module needs to be "implemented", as described in Section 5.6.5 of [RFC7950], in order for the identities in Section 2.1.2 to be defined.

2.2. Example Usage

This following example illustrates how the "transport-params-grouping' grouping appears when populated with some data.

```
<transport-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
  <host-key>
    <host-key-alg>algs:x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>algs:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      algs:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>algs:aes256-ctr</encryption-alg>
    <encryption-alg>algs:aes192-ctr</encryption-alg>
    <encryption-alg>algs:aes128-ctr</encryption-alg>
    <encryption-alg>algs:aes256-cbc</encryption-alg>
    <encryption-alg>algs:aes192-cbc</encryption-alg>
    <encryption-alg>algs:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>algs:hmac-sha2-256</mac-alg>
    <mac-alg>algs:hmac-sha2-512</mac-alg>
  </mac>
</transport-params>
```

2.3. YANG Module

This YANG module has normative references to [RFC4253], [RFC4344], [RFC4419], [RFC5656], [RFC6187], and [RFC6668].

```
<CODE BEGINS> file "ietf-ssh-common@2020-07-10.yang"

module ietf-ssh-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix sshcmn;

  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 <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

      description
        "This module defines a common features, identities, and
         groupings for Secure Shell (SSH).

         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 EEEE
         (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
         itself for full legal notices.;

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

      revision 2020-07-10 {
        description
          "Initial version";
        reference
          "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
      }

      // Features

      feature ssh-ecc {
        description
          "Elliptic Curve Cryptography is supported for SSH.";
        reference
          "RFC 5656: Elliptic Curve Algorithm Integration in the
                     Secure Shell Transport Layer";
      }

      feature ssh-x509-certs {
        description
```

```
        "X.509v3 certificates are supported for SSH per RFC 6187.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
                   Authentication";
    }

    feature ssh-dh-group-exchange {
      description
        "Diffie-Hellman Group Exchange is supported for SSH.";
      reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    feature ssh-ctr {
      description
        "SDCTR encryption mode is supported for SSH.";
      reference
        "RFC 4344: The Secure Shell (SSH) Transport Layer
                   Encryption Modes";
    }

    feature ssh-sha2 {
      description
        "The SHA2 family of cryptographic hash functions is
         supported for SSH.";
      reference
        "FIPS PUB 180-4: Secure Hash Standard (SHS)";
    }

    // Identities

    identity public-key-alg-base {
      description
        "Base identity used to identify public key algorithms.";
    }

    identity ssh-dss {
      base public-key-alg-base;
      description
        "Digital Signature Algorithm using SHA-1 as the
         hashing algorithm.";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity ssh-rsa {
```

```
      base public-key-alg-base;
      description
        "RSASSA-PKCS1-v1_5 signature scheme using SHA-1 as the
         hashing algorithm.";
      reference
        "RFC 4253:
           The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity ecdsa-sha2-nistp256 {
      if-feature "ssh-ecc and ssh-sha2";
      base public-key-alg-base;
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp256 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-nistp384 {
      if-feature "ssh-ecc and ssh-sha2";
      base public-key-alg-base;
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp384 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-nistp521 {
      if-feature "ssh-ecc and ssh-sha2";
      base public-key-alg-base;
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp521 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity x509v3-ssh-rsa {
      if-feature "ssh-x509-certs";
      base public-key-alg-base;
      description
        "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
         in an X.509v3 certificate and using SHA-1 as the hashing
```

```
          algorithm.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
                   Authentication";
    }

    identity x509v3-rsa2048-sha256 {
      if-feature "ssh-x509-certs and ssh-sha2";
      base public-key-alg-base;
      description
        "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
         in an X.509v3 certificate and using SHA-256 as the hashing
         algorithm.  RSA keys conveyed using this format MUST have a
         modulus of at least 2048 bits.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
                   Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp256 {
      if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
      base public-key-alg-base;
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA)
         using the nistp256 curve with a public key stored in
         an X.509v3 certificate and using the SHA2 family of
         hashing algorithms.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
                   Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp384 {
      if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
      base public-key-alg-base;
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA)
         using the nistp384 curve with a public key stored in
         an X.509v3 certificate and using the SHA2 family of
         hashing algorithms.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
                   Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp521 {
      if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
      base public-key-alg-base;
```

```
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA)
         using the nistp521 curve with a public key stored in
         an X.509v3 certificate and using the SHA2 family of
         hashing algorithms.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
                   Authentication";
    }

    identity key-exchange-alg-base {
      description
        "Base identity used to identify key exchange algorithms.";
    }

    identity diffie-hellman-group14-sha1 {
      base key-exchange-alg-base;
      description
        "Diffie-Hellman key exchange with SHA-1 as HASH and
         Oakley Group 14 (2048-bit MODP Group).";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity diffie-hellman-group-exchange-sha1 {
      if-feature "ssh-dh-group-exchange";
      base key-exchange-alg-base;
      description
        "Diffie-Hellman Group and Key Exchange with SHA-1 as HASH.";
      reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    identity diffie-hellman-group-exchange-sha256 {
      if-feature "ssh-dh-group-exchange and ssh-sha2";
      base key-exchange-alg-base;
      description
        "Diffie-Hellman Group and Key Exchange with SHA-256 as HASH.";
      reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    identity ecdh-sha2-nistp256 {
      if-feature "ssh-ecc and ssh-sha2";
      base key-exchange-alg-base;
      description
```

```
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
         nistp256 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdh-sha2-nistp384 {
      if-feature "ssh-ecc and ssh-sha2";
      base key-exchange-alg-base;
      description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
         nistp384 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdh-sha2-nistp521 {
      if-feature "ssh-ecc and ssh-sha2";
      base key-exchange-alg-base;
      description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
         nistp521 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity encryption-alg-base {
      description
        "Base identity used to identify encryption algorithms.";
    }

    identity triple-des-cbc {
      base encryption-alg-base;
      description
        "Three-key 3DES in CBC mode.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity aes128-cbc {
      base encryption-alg-base;
      description
        "AES in CBC mode, with a 128-bit key.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
```

```
  }

  identity aes192-cbc {
    base encryption-alg-base;
    description
      "AES in CBC mode, with a 192-bit key.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity aes256-cbc {
    base encryption-alg-base;
    description
      "AES in CBC mode, with a 256-bit key.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity aes128-ctr {
    if-feature "ssh-ctr";
    base encryption-alg-base;
    description
      "AES in SDCTR mode, with 128-bit key.";
    reference
      "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
                 Modes";
  }

  identity aes192-ctr {
    if-feature "ssh-ctr";
    base encryption-alg-base;
    description
      "AES in SDCTR mode, with 192-bit key.";
    reference
      "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
                 Modes";
  }

  identity aes256-ctr {
    if-feature "ssh-ctr";
    base encryption-alg-base;
    description
      "AES in SDCTR mode, with 256-bit key.";
    reference
      "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
         Modes";
  }
```

```
  identity mac-alg-base {
    description
      "Base identity used to identify message authentication
       code (MAC) algorithms.";
  }

  identity hmac-sha1 {
    base mac-alg-base;
    description
      "HMAC-SHA1";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity hmac-sha2-256 {
    if-feature "ssh-sha2";
    base mac-alg-base;
    description
      "HMAC-SHA2-256";
    reference
      "RFC 6668: SHA-2 Data Integrity Verification for the
                 Secure Shell (SSH) Transport Layer Protocol";
  }

  identity hmac-sha2-512 {
    if-feature "ssh-sha2";
    base mac-alg-base;
    description
      "HMAC-SHA2-512";
    reference
      "RFC 6668: SHA-2 Data Integrity Verification for the
                 Secure Shell (SSH) Transport Layer Protocol";
  }

  // Groupings

  grouping transport-params-grouping {
    description
      "A reusable grouping for SSH transport parameters.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    container host-key {
      description
        "Parameters regarding host key.";
      leaf-list host-key-alg {
        type identityref {
          base public-key-alg-base;
        }
```

```
          ordered-by user;
          description
            "Acceptable host key algorithms in order of descending
             preference.  The configured host key algorithms should
             be compatible with the algorithm used by the configured
             private key.  Please see Section 5 of RFC EEEE for
             valid combinations.

             If this leaf-list is not configured (has zero elements)
             the acceptable host key algorithms are implementation-
             defined.";
          reference
            "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
        }
      }
      container key-exchange {
        description
          "Parameters regarding key exchange.";
        leaf-list key-exchange-alg {
          type identityref {
            base key-exchange-alg-base;
          }
          ordered-by user;
          description
            "Acceptable key exchange algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable key exchange algorithms are implementation
             defined.";
        }
      }
      container encryption {
        description
          "Parameters regarding encryption.";
        leaf-list encryption-alg {
          type identityref {
            base encryption-alg-base;
          }
          ordered-by user;
          description
            "Acceptable encryption algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable encryption algorithms are implementation
             defined.";
        }
```

```
        }
        container mac {
          description
            "Parameters regarding message authentication code (MAC).";
          leaf-list mac-alg {
            type identityref {
              base mac-alg-base;
            }
            ordered-by user;
            description
              "Acceptable MAC algorithms in order of descending
               preference.

               If this leaf-list is not configured (has zero elements)
               the acceptable MAC algorithms are implementation-
               defined.";
          }
        }
      }
    }

    <CODE ENDS>
```

3. The "ietf-ssh-client" Module

3.1. Data Model Overview

3.1.1. Features

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

```
Features:
  +-- ssh-client-transport-params-config
  +-- ssh-client-keepalives
  +-- client-identity-password
  +-- client-identity-publickey
  +-- client-identity-hostbased
  +-- client-identity-none
```

3.1.2. Groupings

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

```
Groupings:
  +-- ssh-client-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "ssh-client-grouping" Grouping

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

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

  grouping ssh-client-grouping
    +-- client-identity
    |  +-- username?      string
    |  +-- public-key! {client-identity-publickey}?
    |  |  +---u ks:local-or-keystore-asymmetric-key-grouping
    |  +-- password?      string {client-identity-password}?
    |  +-- hostbased! {client-identity-hostbased}?
    |  |  +---u ks:local-or-keystore-asymmetric-key-grouping
    |  +-- none?          empty {client-identity-none}?
    |  +-- certificate! {sshcmn:ssh-x509-certs}?
    |     +---u ks:local-or-keystore-end-entity-cert-with-key-groupi\
ng
    +-- server-authentication
    |  +-- ssh-host-keys!
    |  |  +---u ts:local-or-truststore-public-keys-grouping
    |  +-- ca-certs! {sshcmn:ssh-x509-certs}?
    |  |  +---u ts:local-or-truststore-certs-grouping
    |  +-- ee-certs! {sshcmn:ssh-x509-certs}?
    |     +---u ts:local-or-truststore-certs-grouping
    +-- transport-params {ssh-client-transport-params-config}?
    |  +---u sshcmn:transport-params-grouping
    +-- keepalives! {ssh-client-keepalives}?
       +-- max-wait?       uint16
       +-- max-attempts?   uint8
```

Comments:

* The "client-identity" node configures a "username" and credentials, each enabled by a "feature" statement defined in Section 3.1.1.

* The "server-authentication" node configures trust anchors for authenticating the SSH server, with each option enabled by a "feature" statement.

* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH server. The aliveness-test occurs at the SSH protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "transport-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

3.2. Example Usage

This section presents two examples showing the "ssh-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the client identity and server authentication:

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

<ssh-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <username>foobar</username>
    <public-key>
      <local-definition>
        <public-key-format>ct:ssh-public-key-format</public-key-form\
\at>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-f\
```

```
   \ormat>
           <cleartext-private-key>base64encodedvalue==</cleartext-priva\
   \te-key>
         </local-definition>
       </public-key>
     </client-identity>

     <!-- which host keys will this client trust -->
     <server-authentication>
       <ssh-host-keys>
         <local-definition> <!-- FIXME: float 'local-def' down to each?\
   \ -->
           <!--<ssh-public-key>-->
           <public-key>
             <name>corp-fw1</name>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   \rmat>
             <public-key>base64encodedvalue==</public-key>
             <!--
           </ssh-public-key>
           <ssh-public-key>
           -->
           </public-key>
           <public-key>
             <name>corp-fw2</name>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   \rmat>
             <public-key>base64encodedvalue==</public-key>
             <!--</ssh-public-key>-->
           </public-key>
         </local-definition>
       </ssh-host-keys>
       <ca-certs>
         <local-definition>
           <certificate>
             <name>Server Cert Issuer #1</name>
             <cert-data>base64encodedvalue==</cert-data>
           </certificate>
           <certificate>
             <name>Server Cert Issuer #2</name>
             <cert-data>base64encodedvalue==</cert-data>
           </certificate>
         </local-definition>
       </ca-certs>
       <ee-certs>
         <local-definition>
           <certificate>
             <name>My Application #1</name>
```

```
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>My Application #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </local-definition>
      </ee-certs>
    </server-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-client>
```

The following configuration example uses keystore-references for the client identity and truststore-references for server authentication: from the keystore:

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

   <ssh-client
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
     xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

     <!-- how this client will authenticate itself to the server -->
     <client-identity>
       <username>foobar</username>
       <!-- can an SSH client have move than one key?
       <public-key>
         <keystore-reference>ssh-rsa-key</keystore-reference>
       </public-key>
       -->
       <certificate>
         <keystore-reference>
           <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
           <certificate>ex-rsa-cert2</certificate>
         </keystore-reference>
       </certificate>
     </client-identity>

     <!-- which host-keys will this client trust -->
     <server-authentication>
       <ssh-host-keys> <!-- FIXME: should 'ts-ref' be to bag or each ke\
   y? -->
         <truststore-reference>trusted-ssh-public-keys</truststore-refe\
   rence>
       </ssh-host-keys>
       <ca-certs> <!-- FIXME: should 'ts-ref' be to bag or each key? -->
         <truststore-reference>trusted-server-ca-certs</truststore-refe\
   rence>
       </ca-certs>
       <ee-certs> <!-- FIXME: should 'ts-ref' be to bag or each key? -->
         <truststore-reference>trusted-server-ee-certs</truststore-refe\
   rence>
       </ee-certs>
     </server-authentication>

     <keepalives>
       <max-wait>30</max-wait>
       <max-attempts>3</max-attempts>
     </keepalives>

   </ssh-client>
```

3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors], and [I-D.ietf-netconf-keystore].

<CODE BEGINS> file "ietf-ssh-client@2020-07-10.yang"

```
module ietf-ssh-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
  prefix sshc;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2020-07-10; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
       Author:   Gary Wu <mailto:garywu@cisco.com>";

     description
       "This module defines reusable groupings for SSH clients that
        can be used as a basis for specific SSH client instances.

        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 EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
        itself for full legal notices.;

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

     revision 2020-07-10 {
       description
         "Initial version";
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }

     // Features

     feature ssh-client-transport-params-config {
       description
         "SSH transport layer parameters are configurable on an SSH
          client.";
     }

     feature ssh-client-keepalives {
       description
         "Per socket SSH keepalive parameters are configurable for
          SSH clients on the server implementing this feature.";
```

```
  }

  feature client-identity-password {
    description
      "Indicates that the 'password' authentication type
       is supported for client identification.";
  }

  feature client-identity-publickey {
    description
      "Indicates that the 'publickey' authentication type
       is supported for client identification.

       The 'publickey' authentication type is required by
       RFC 4252, but common implementations enable it to
       be disabled.";
  }

  feature client-identity-hostbased {
    description
      "Indicates that the 'hostbased' authentication type
       is supported for client identification.";
  }

  feature client-identity-none {
    description
      "Indicates that the 'none' authentication type is
       supported for client identification.";
  }

  // Groupings

  grouping ssh-client-grouping {
    description
      "A reusable grouping for configuring a SSH client without
       any consideration for how an underlying TCP session is
       established.

       Note that this grouping uses fairly typical descendent
       node names such that a stack of 'uses' statements will
       have name conflicts.  It is intended that the consuming
       data model will resolve the issue (e.g., by wrapping
       the 'uses' statement in a container called
       'ssh-client-parameters').  This model purposely does
       not do this itself so as to provide maximum flexibility
       to consuming models.";

    container client-identity {
```

```
        nacm:default-deny-write;
        must
          'public-key or password or hostbased or none or certificate';
        description
          "The credentials that the client may use, pending
           the SSH server's requirements, by the SSH client
           to authenticate to the SSH server.";
        leaf username {
          type string;
          description
            "The username of this user.  This will be the username
             used, for instance, to log into an SSH server.";
        }
        container public-key {
          if-feature client-identity-publickey;
          presence
            "Indicates that publickey-based authentication
             is configured";
          description
            "A locally-defined or referenced asymmetric key
             pair to be used for client identification.";
          reference
            "RFC CCCC: A YANG Data Model for a Keystore";
          uses ks:local-or-keystore-asymmetric-key-grouping {
            refine "local-or-keystore/local/local-definition" {
              must 'public-key-format = "ct:ssh-public-key-format"';
            }
            refine "local-or-keystore/keystore/keystore-reference" {
              must 'deref(.)/../ks:public-key-format'
                   + ' = "ct:ssh-public-key-format"';
            }
          }
        }
        leaf password {
          if-feature client-identity-password;
          nacm:default-deny-all;
          type string;
          description
            "A password to be used for client identification.";
        }
        container hostbased {
          if-feature client-identity-hostbased;
          presence
            "Indicates that hostbased authentication is configured";
          description
            "A locally-defined or referenced asymmetric key
             pair to be used for host identification.";
          reference
```

```
            "RFC CCCC: A YANG Data Model for a Keystore";
          uses ks:local-or-keystore-asymmetric-key-grouping {
            refine "local-or-keystore/local/local-definition" {
              must 'public-key-format = "ct:ssh-public-key-format"';
            }
            refine "local-or-keystore/keystore/keystore-reference" {
              must 'deref(.)/../ks:public-key-format'
                   + ' = "ct:ssh-public-key-format"';
            }
          }
        }
        leaf none {
          if-feature client-identity-none;
          type empty;
          description
            "Indicates that 'none' algorithm is used for client
             identification.";
        }
        container certificate {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that certificate-based authentication
             is configured";
          description
            "A locally-defined or referenced certificate
             to be used for client identification.";
          reference
            "RFC CCCC: A YANG Data Model for a Keystore";
          uses
          ks:local-or-keystore-end-entity-cert-with-key-grouping {
            refine "local-or-keystore/local/local-definition" {
              must
                'public-key-format'
                + ' = "ct:subject-public-key-info-format"';
            }
            refine "local-or-keystore/keystore/keystore-reference"
                   + "/asymmetric-key" {
              must 'deref(.)/../ks:public-key-format'
                   + ' = "ct:subject-public-key-info-format"';
            }
          }
        }
      } // container client-identity

      container server-authentication {
        nacm:default-deny-write;
        must 'ssh-host-keys or ca-certs or ee-certs';
        description
```

```
          "Specifies how the SSH client can authenticate SSH servers.
           Any combination of credentials is additive and unordered.";
        container ssh-host-keys {
          presence
            "Indicates that the client can authenticate servers
             using the configured SSH host keys.";
          description
            "A list of SSH host keys used by the SSH client to
             authenticate SSH server host keys.  A server host key
             is authenticated if it is an exact match to a
             configured SSH host key.";
          reference
             "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine
              "local-or-truststore/local/local-definition/public-key" {
              must 'public-key-format = "ct:ssh-public-key-format"';
            }
            refine
              "local-or-truststore/truststore/truststore-reference" {
              must 'deref(.)/../*/ts:public-key-format'
                    + ' = "ct:ssh-public-key-format"';
            }
          }
        }
        container ca-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that the client can authenticate servers
             using the configured trust anchor certificates.";
          description
            "A set of certificate authority (CA) certificates used by
             the SSH client to authenticate SSH servers.  A server
             is authenticated if its certificate  has a valid chain
             of trust to a configured CA certificate.";
           reference
             "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that the client can authenticate servers
             using the configured end-entity certificates.";
          description
            "A set of end-entity certificates used by the SSH client
             to authenticate SSH servers.  A server is authenticated
             if its certificate is an exact match to a configured
```

```
            end-entity certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
        uses ts:local-or-truststore-certs-grouping;
      }
    } // container server-authentication

    container transport-params {
      nacm:default-deny-write;
      if-feature "ssh-client-transport-params-config";
      description
        "Configurable parameters of the SSH transport layer.";
      uses sshcmn:transport-params-grouping;
    } // container transport-parameters

    container keepalives {
      nacm:default-deny-write;
      if-feature "ssh-client-keepalives";
      presence
        "Indicates that the SSH client proactively tests the
         aliveness of the remote SSH server.";
      description
        "Configures the keep-alive policy, to proactively test
         the aliveness of the SSH server.  An unresponsive TLS
         server is dropped after approximately max-wait *
         max-attempts seconds.  Per Section 4 of RFC 4254,
         the SSH client SHOULD send an SSH_MSG_GLOBAL_REQUEST
         message with a purposely nonexistent 'request name'
         value (e.g., keepalive@ietf.org) and the 'want reply'
         value set to '1'.";
      reference
        "RFC 4254: The Secure Shell (SSH) Connection Protocol";
      leaf max-wait {
        type uint16 {
          range "1..max";
        }
        units "seconds";
        default "30";
        description
          "Sets the amount of time in seconds after which if
           no data has been received from the SSH server, a
           TLS-level message will be sent to test the
           aliveness of the SSH server.";
      }
      leaf max-attempts {
        type uint8;
        default "3";
        description
```

```
            "Sets the maximum number of sequential keep-alive
             messages that can fail to obtain a response from
             the SSH server before assuming the SSH server is
             no longer alive.";
          }
        } // container keepalives
      } // grouping ssh-client-grouping
    } // module ietf-ssh-client
```

<CODE ENDS>

4. The "ietf-ssh-server" Module

4.1. Data Model Overview

4.1.1. Features

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

```
Features:
  +-- ssh-server-transport-params-config
  +-- ssh-server-keepalives
  +-- client-auth-config-supported
  +-- client-auth-publickey
  +-- client-auth-password
  +-- client-auth-hostbased
  +-- client-auth-none
```

4.1.2. Groupings

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

```
Groupings:
  +-- ssh-server-grouping
```

Each of these groupings are presented in the following subsections.

4.1.2.1. The "ssh-server-grouping" Grouping

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

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

     grouping ssh-server-grouping
       +-- server-identity
       |  +-- host-key* [name]
       |     +-- name?                string
       |     +-- (host-key-type)
       |        +--:(public-key)
       |        |  +-- public-key
       |        |     +---u ks:local-or-keystore-asymmetric-key-grouping
       |        +--:(certificate)
       |           +-- certificate {sshcmn:ssh-x509-certs}?
       |              +---u ks:local-or-keystore-end-entity-cert-with-k\
   ey-grouping
       +-- client-authentication
       |  +-- supported-authentication-methods
       |  |  +-- publickey?   empty
       |  |  +-- password?    empty {client-auth-password}?
       |  |  +-- hostbased?   empty {client-auth-hostbased}?
       |  |  +-- none?        empty {client-auth-none}?
       |  +-- users {client-auth-config-supported}?
       |  |  +-- user* [name]
       |  |     +-- name?          string
       |  |     +-- public-keys! {client-auth-publickey}?
       |  |     |  +---u ts:local-or-truststore-public-keys-grouping
       |  |     +-- password?      ianach:crypt-hash
       |  |     |       {client-auth-password}?
       |  |     +-- hostbased! {client-auth-hostbased}?
       |  |     |  +---u ts:local-or-truststore-public-keys-grouping
       |  |     +-- none?          empty {client-auth-none}?
       |  +-- ca-certs!
       |  |       {client-auth-config-supported,sshcmn:ssh-x509-certs}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- ee-certs!
       |          {client-auth-config-supported,sshcmn:ssh-x509-certs}?
       |     +---u ts:local-or-truststore-certs-grouping
       +-- transport-params {ssh-server-transport-params-config}?
       |  +---u sshcmn:transport-params-grouping
       +-- keepalives! {ssh-server-keepalives}?
          +-- max-wait?       uint16
          +-- max-attempts?   uint8
```

Comments:

* The "server-identity" node configures identity credentials. The ability to use a certificate is enabled by a "feature".

* The "client-authentication" node configures trust anchors for authenticating the SSH client, with each option enabled by a "feature" statement.

* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH client. The aliveness-test occurs at the SSH protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "transport-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

4.2. Example Usage

This section presents two examples showing the "ssh-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the server identity and client authentication:

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

<ssh-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- the host-key this SSH server will present -->
```

```
     <server-identity>
       <host-key>
         <name>my-pubkey-based-host-key</name>
         <public-key>
           <local-definition>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
rmat>
             <public-key>base64encodedvalue==</public-key>
             <private-key-format>ct:rsa-private-key-format</private-key\
-format>
             <cleartext-private-key>base64encodedvalue==</cleartext-pri\
vate-key>
           </local-definition>
         </public-key>
       </host-key>
       <host-key>
         <name>my-cert-based-host-key</name>
         <certificate>
           <local-definition>
             <public-key-format>ct:subject-public-key-info-format</publ\
ic-key-format>
             <public-key>base64encodedvalue==</public-key>
             <private-key-format>ct:rsa-private-key-format</private-key\
-format>
             <cleartext-private-key>base64encodedvalue==</cleartext-pri\
vate-key>
             <cert-data>base64encodedvalue==</cert-data>
           </local-definition>
         </certificate>
       </host-key>
     </server-identity>

     <!-- the client credentials this SSH server will trust -->
     <client-authentication>
       <supported-authentication-methods>
         <publickey/>
       </supported-authentication-methods>
       <users>
         <user>
           <name>mary</name>
           <password>$0$secret</password>
           <public-keys>
             <local-definition>
               <!--<ssh-public-key>-->
               <public-key>
                 <name>User A</name>
                 <public-key-format>ct:ssh-public-key-format</public-ke\
y-format>
```

```
                <public-key>base64encodedvalue==</public-key>
                <!--</ssh-public-key>
              <ssh-public-key>-->
              </public-key>
              <public-key>
                <name>User B</name>
                <public-key-format>ct:ssh-public-key-format</public-ke\
   y-format>
                <public-key>base64encodedvalue==</public-key>
              </public-key>
              <!--</ssh-public-key>-->
            </local-definition>
          </public-keys>
        </user>
      </users>
      <ca-certs>
        <local-definition>
          <certificate>
            <name>Identity Cert Issuer #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Identity Cert Issuer #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </local-definition>
      </ca-certs>
      <ee-certs>
        <local-definition>
          <certificate>
            <name>Application #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Application #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </local-definition>
      </ee-certs>
    </client-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-server>
```

The following configuration example uses keystore-references for the server identity and truststore-references for client authentication: from the keystore:

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

<ssh-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- the host-key this SSH server will present -->
  <server-identity>
    <host-key>
      <name>my-pubkey-based-host-key</name>
      <public-key>
        <keystore-reference>ssh-rsa-key</keystore-reference>
      </public-key>
    </host-key>
    <host-key>
      <name>my-cert-based-host-key</name>
      <certificate>
        <keystore-reference>
          <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
          <certificate>ex-rsa-cert2</certificate>
        </keystore-reference>
      </certificate>
    </host-key>
  </server-identity>

  <!-- the client credentials this SSH server will trust -->
  <client-authentication>
    <supported-authentication-methods>
      <publickey/>
    </supported-authentication-methods>
    <users>
      <user>
        <name>mary</name>
        <password>$0$secret</password>
        <public-keys>
          <truststore-reference>SSH Public Keys for Application A</t\
truststore-reference>
        </public-keys>
      </user>
    </users>
    <ca-certs>
      <truststore-reference>trusted-client-ca-certs</truststore-refe\
rence>
    </ca-certs>
```

```
      <ee-certs>
        <truststore-reference>trusted-client-ee-certs</truststore-refe\
  rence>
      </ee-certs>
    </client-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-server>
```

4.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore] and informative references to [RFC4253] and [RFC7317].

<CODE BEGINS> file "ietf-ssh-server@2020-07-10.yang"

```
module ietf-ssh-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
  prefix sshs;

  import iana-crypt-hash {
    prefix ianach;
    reference
      "RFC 7317: A YANG Data Model for System Management";
  }

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

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

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

```
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2020-07-10; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
     Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module defines reusable groupings for SSH servers that
     can be used as a basis for specific SSH server instances.

     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 EEEE
     (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
     itself for full legal notices.;

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

```
    revision 2020-07-10 {
      description
        "Initial version";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Features

    feature ssh-server-transport-params-config {
      description
        "SSH transport layer parameters are configurable on an SSH
         server.";
    }

    feature ssh-server-keepalives {
      description
        "Per socket SSH keepalive parameters are configurable for
         SSH servers on the server implementing this feature.";
    }

    feature client-auth-config-supported {
      description
        "Indicates that the configuration for how to authenticate
         clients can be configured herein, as opposed to in an
         application specific location.  That is, to support the
         consuming data models that prefer to place client
         authentication with client definitions, rather then
         in a data model principally concerned with configuring
         the transport.";
    }

    feature client-auth-publickey {
      description
        "Indicates that the 'publickey' authentication type
         is supported.

         The 'publickey' authentication type is required by
         RFC 4252, but common implementations enable it to
         be disabled.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    feature client-auth-password {
      description
        "Indicates that the 'password' authentication type
```

```
          is supported.";
      }

      feature client-auth-hostbased {
        description
          "Indicates that the 'hostbased' authentication type
           is supported.";
      }

      feature client-auth-none {
        description
          "Indicates that the 'none' authentication type is
           supported.";
      }

      // Groupings

      grouping ssh-server-grouping {
        description
          "A reusable grouping for configuring a SSH server without
           any consideration for how underlying TCP sessions are
           established.

           Note that this grouping uses fairly typical descendent
           node names such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue (e.g., by wrapping
           the 'uses' statement in a container called
           'ssh-server-parameters').  This model purposely does
           not do this itself so as to provide maximum flexibility
           to consuming models.";

        container server-identity {
          nacm:default-deny-write;
          description
            "The list of host keys the SSH server will present when
             establishing a SSH connection.";
          list host-key {
            key "name";
            min-elements 1;
            ordered-by user;
            description
              "An ordered list of host keys the SSH server will use to
               construct its ordered list of algorithms, when sending
               its SSH_MSG_KEXINIT message, as defined in Section 7.1
               of RFC 4253.";
            reference
              "RFC 4253: The Secure Shell (SSH) Transport Layer
```

```
                Protocol";
          leaf name {
            type string;
            description
              "An arbitrary name for this host key";
          }
          choice host-key-type {
            mandatory true;
            description
              "The type of host key being specified";
            container public-key {
              description
                "A locally-defined or referenced asymmetric key pair
                 to be used for the SSH server's host key.";
              reference
                "RFC CCCC: A YANG Data Model for a Keystore";
              uses ks:local-or-keystore-asymmetric-key-grouping {
                refine "local-or-keystore/local/local-definition" {
                  must
                    'public-key-format = "ct:ssh-public-key-format"';
                }
                refine "local-or-keystore/keystore/"
                       + "keystore-reference" {
                  must 'deref(.)/../ks:public-key-format'
                       + ' = "ct:ssh-public-key-format"';
                }
              }
            }
            container certificate {
              if-feature "sshcmn:ssh-x509-certs";
              description
                "A locally-defined or referenced end-entity
                 certificate to be used for the SSH server's
                 host key.";
              reference
                "RFC CCCC: A YANG Data Model for a Keystore";
              uses
              ks:local-or-keystore-end-entity-cert-with-key-grouping {
                refine "local-or-keystore/local/local-definition" {
                  must
                    'public-key-format'
                    + ' = "ct:subject-public-key-info-format"';
                }
                refine "local-or-keystore/keystore/keystore-reference"
                       + "/asymmetric-key" {
                  must 'deref(.)/../ks:public-key-format'
                       + ' = "ct:subject-public-key-info-format"';
                }
```

```
            }
          }
        }
      }
    } // container server-identity

    container client-authentication {
      nacm:default-deny-write;
      description
        "Specifies how the SSH server can authenticate SSH clients.";
      container supported-authentication-methods {
        description
          "Indicates which authentication methods the server
           supports.";
        leaf publickey {
          type empty;
          description
            "Indicates that the 'publickey' method is supported.
             Note that RFC 6187 X.509v3 Certificates for SSH uses
             the 'publickey' method name.";
          reference
            "RFC 4252: The Secure Shell (SSH) Authentication
                       Protocol.
             RFC 6187: X.509v3 Certificates for Secure Shell
                       Authentication.";
        }
        leaf password {
          if-feature client-auth-password;
          type empty;
          description
            "Indicates that the 'password' method is supported.";
          reference
            "RFC 4252: The Secure Shell (SSH) Authentication
                       Protocol.";
        }
        leaf hostbased {
          if-feature client-auth-hostbased;
          type empty;
          description
            "Indicates that the 'hostbased' method is supported.";
          reference
            "RFC 4252: The Secure Shell (SSH) Authentication
                       Protocol.";
        }
        leaf none {
          if-feature client-auth-none;
          type empty;
          description
```

```
            "Indicates that the 'none' method is supported.";
          reference
            "RFC 4252: The Secure Shell (SSH) Authentication
                       Protocol.";
        }
      }

      container users {
        if-feature "client-auth-config-supported";
        description
          "A list of locally configured users.";
        list user {
          key name;
          description
            "The list of local users configured on this device.";
          leaf name {
            type string;
            description
             "The user name string identifying this entry.";
          }
          container public-keys {
            if-feature client-auth-publickey;
            presence
              "Indicates that the server can authenticate this
               user using any of the configured SSH public keys.";
            description
              "A set of SSH public keys may be used by the SSH
               server to authenticate this user.  A user is
               authenticated if its public key is an exact
               match to a configured public key.";
            reference
              "RFC BBBB: A YANG Data Model for a Truststore";
            uses ts:local-or-truststore-public-keys-grouping {
              refine "local-or-truststore/local/local-definition"
                     + "/public-key" {
                must 'public-key-format'
                     + ' = "ct:ssh-public-key-format"';
              }
              refine "local-or-truststore/truststore/"
                     + "truststore-reference" {
                must 'deref(.)/../*/ts:public-key-format'
                     + ' = "ct:ssh-public-key-format"';
              }
            }
          }
          leaf password {
            if-feature client-auth-password;
            type ianach:crypt-hash;
```

```
            description
              "The password for this user.";
          }

          container hostbased {
            if-feature client-auth-hostbased;
            presence
              "Indicates that the server can authenticate this
               user's 'host' using any of the configured SSH
               host keys.";
            description
              "A set of SSH host keys may be used by the SSH
               server to authenticate this user's host.  A
               user's host is authenticated if its host key
               is an exact match to a configured host key.";
            reference
              "RFC 4253: The Secure Shell (SSH) Transport Layer
               RFC BBBB: A YANG Data Model for a Truststore";
            uses ts:local-or-truststore-public-keys-grouping {
              refine "local-or-truststore/local/local-definition"
                     + "/public-key" {
                must 'public-key-format'
                     + ' = "ct:ssh-public-key-format"';
              }
              refine "local-or-truststore/truststore"
                     + "/truststore-reference" {
                must 'deref(.)/../*/ts:public-key-format'
                     + ' = "ct:ssh-public-key-format"';
              }
            }
          }
          leaf none {
            if-feature client-auth-none;
            type empty;
            description
              "Indicates that the 'none' method is supported.";
            reference
              "RFC 4252: The Secure Shell (SSH) Authentication
                         Protocol.";
          }
        }
      }
      container ca-certs {
        if-feature "client-auth-config-supported";
        if-feature "sshcmn:ssh-x509-certs";
        presence
          "Indicates that the SSH server can authenticate SSH
           clients using configured certificate authority (CA)
```

```
            certificates.";
          description
            "A set of certificate authority (CA) certificates used by
             the SSH server to authenticate SSH client certificates.
             A client certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "client-auth-config-supported";
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that the SSH server can authenticate SSH
             clients using configured end-entity certificates.";
          description
            "A set of client certificates (i.e., end entity
             certificates) used by the SSH server to authenticate
             the certificates presented by SSH clients.  A client
             certificate is authenticated if it is an exact match
             to a configured end-entity certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
      } // container client-authentication

      container transport-params {
        nacm:default-deny-write;
        if-feature "ssh-server-transport-params-config";
        description
          "Configurable parameters of the SSH transport layer.";
        uses sshcmn:transport-params-grouping;
      } // container transport-params

      container keepalives {
        nacm:default-deny-write;
        if-feature "ssh-server-keepalives";
        presence
          "Indicates that the SSH server proactively tests the
           aliveness of the remote SSH client.";
        description
          "Configures the keep-alive policy, to proactively test
           the aliveness of the SSL client.  An unresponsive SSL
           client is dropped after approximately max-wait *
           max-attempts seconds.  Per Section 4 of RFC 4254,
           the SSH server SHOULD send an SSH_MSG_GLOBAL_REQUEST
```

```
            message with a purposely nonexistent 'request name'
            value (e.g., keepalive@ietf.org) and the 'want reply'
            value set to '1'.";
        reference
          "RFC 4254: The Secure Shell (SSH) Connection Protocol";
        leaf max-wait {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default "30";
          description
            "Sets the amount of time in seconds after which
             if no data has been received from the SSL client,
             a SSL-level message will be sent to test the
             aliveness of the SSL client.";
        }
        leaf max-attempts {
          type uint8;
          default "3";
          description
            "Sets the maximum number of sequential keep-alive
             messages that can fail to obtain a response from
             the SSL client before assuming the SSL client is
             no longer alive.";
        }
      }
    } // grouping ssh-server-grouping
  } // module ietf-ssh-server

  <CODE ENDS>
```

5. Security Considerations

5.1. The "ietf-ssh-common" YANG Module

The "ietf-ssh-common" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. The "ietf-ssh-client" YANG Module

The "ietf-ssh-client" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "client-identity/password" node:

  The cleartext "password" node defined in the "ssh-client-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All of the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.3. The "ietf-ssh-server" YANG Module

The "ietf-ssh-server" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All of the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., or even the modification of transport or keepalive parameters can dramatically alter the

implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers three URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-ssh-common
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-client
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-server
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

6.2. The "YANG Module Names" Registry

This document registers three YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-ssh-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-common
prefix:       sshcmn
reference:    RFC EEEE

name:         ietf-ssh-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-client
prefix:       sshc
reference:    RFC EEEE

name:         ietf-ssh-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-server
prefix:       sshs
reference:    RFC EEEE
```

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

[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>.

[RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <https://www.rfc-editor.org/info/rfc4344>.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <https://www.rfc-editor.org/info/rfc4419>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[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>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <https://www.rfc-editor.org/info/rfc6668>.

[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>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

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

[OPENSSH] Project, T. O., "OpenSSH", 2016, <http://www.openssh.com>.

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

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, DOI 10.17487/RFC4254, January 2006, <https://www.rfc-editor.org/info/rfc4254>.

[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>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

Appendix A. Change Log

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

A.1. 00 to 01

* Noted that '0.0.0.0' and '::' might have special meanings.

* Renamed "keychain" to "keystore".

A.2. 01 to 02

* Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.

* Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.

* Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

A.3. 02 to 03

* Removed 'RESTRICTED' enum from 'password' leaf type.

* Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

* Fixed description statement for leaf 'trusted-ca-certs'.

A.4. 03 to 04

* Change title to "YANG Groupings for SSH Clients and SSH Servers"
* Added reference to RFC 6668
* Added RFC 8174 to Requirements Language Section.
* Enhanced description statement for ietf-ssh-server's "trusted-ca-certs" leaf.
* Added mandatory true to ietf-ssh-client's "client-auth" 'choice' statement.
* Changed the YANG prefix for module ietf-ssh-common from 'sshcom' to 'sshcmn'.
* Removed the compression algorithms as they are not commonly configurable in vendors' implementations.
* Updating descriptions in transport-params-grouping and the servers's usage of it.
* Now tree diagrams reference ietf-netmod-yang-tree-diagrams
* Updated YANG to use typedefs around leafrefs to common keystore paths
* Now inlines key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

* Merged changes from co-author.

A.6. 05 to 06

* Updated to use trust anchors from trust-anchors draft (was keystore draft)
* Now uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

A.7. 06 to 07

* factored the ssh-[client|server]-groupings into more reusable groupings.

* added if-feature statements for the new "ssh-host-keys" and "x509-certificates" features defined in draft-ietf-netconf-trust-anchors.

A.8. 07 to 08

* Added a number of compatibility matrices to Section 5 (thanks Frank!)

* Clarified that any configured "host-key-alg" values need to be compatible with the configured private key.

A.9. 08 to 09

* Updated examples to reflect update to groupings defined in the keystore -09 draft.

* Add SSH keepalives features and groupings.

* Prefixed top-level SSH grouping nodes with 'ssh-' and support mashups.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

* Reformatted the YANG modules.

A.11. 10 to 11

* Reformatted lines causing folding to occur.

A.12. 11 to 12

* Collapsed all the inner groupings into the top-level grouping.

* Added a top-level "demux container" inside the top-level grouping.

* Added NACM statements and updated the Security Considerations section.

* Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

A.13. 12 to 13

* Removed the "demux containers", floating the nacm:default-deny-write to each descendent node, and adding a note to model designers regarding the potential need to add their own demux containers.

* Fixed a couple references (section 2 --> section 3)

* In the server model, replaced <client-cert-auth> with <client-authentication> and introduced 'local-or-external' choice.

A.14. 13 to 14

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

A.15. 14 to 15

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:host-keys-ref" or "ts:certificates-ref" to a container that uses "ts:local-or-truststore-host-keys-grouping" or "ts:local-or-truststore-certs-grouping".

A.16. 15 to 16

* Removed unnecessary if-feature statements in the -client and -server modules.

* Cleaned up some description statements in the -client and -server modules.

* Fixed a canonical ordering issue in ietf-ssh-common detected by new pyang.

A.17. 16 to 17

* Removed choice local-or-external by removing the 'external' case and flattening the 'local' case and adding a "client-auth-config-supported" feature.

* Updated examples to include the "*-key-format" nodes.

* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:ssh-public-key-format" (must expr for ref'ed keys are TBD).

A.18. 17 to 18

* Removed leaf-list 'other' from ietf-ssh-server.

* Removed unused 'external-client-auth-supported' feature.

* Added features client-auth-password, client-auth-hostbased, and client-auth-none.

* Renamed 'host-key' to 'public-key' for when refering to 'publickey' based auth.

* Added new feature-protected 'hostbased' and 'none' to the 'user' node's config.

* Added new feature-protected 'hostbased' and 'none' to the 'client-identity' node's config.

* Updated examples to reflect new "bag" addition to truststore.

* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.

* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags.

* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

A.19. 18 to 19

* Updated the "keepalives" containers to address Michal Vasko's request to align with RFC 8071.

* Removed algorithm-mapping tables from the "SSH Common Model" section

* Removed 'algorithm' node from examples.

* Added feature "client-identity-publickey"

* Removed "choice auth-type", as auth-types aren't exclusive.

* Renamed both "client-certs" and "server-certs" to "ee-certs"

* Switch "must" to assert the public-key-format is "subject-public-key-info-format" when certificates are used.

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

A.20. 19 to 20

* Added a "must 'public-key or password or hostbased or none or certificate'" statement to the "user" node in ietf-ssh-client

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

* Moved the "ietf-ssh-common" module section to proceed the other two module sections.

* Updated the Security Considerations section.

A.21. 20 to 21

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Mehmet Ersue, Balazs Kovacs, Radek Krejci, David Lamparter, Ladislav Lhotka, Alan Luchuk, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Michal Vasko, Bert Wijnen, and Liang Xia.

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Gary Wu
Cisco Systems

Email: garywu@cisco.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-40

Abstract

This document presents seven YANG 1.1 modules. Three IETF modules, and four supporting IANA modules.

The three IETF modules are: ietf-ssh-common, ietf-ssh-client, and ietf-ssh-server. The "ietf-ssh-client" and "ietf-ssh-server" modules are the primary productions of this work, supporting the configuration and monitoring of SSH clients and servers.

The four IANA modules are: iana-ssh-encryption-algs, iana-ssh-key-exchange-algs, iana-ssh-mac-algs, and iana-ssh-public-key-algs. These modules each define YANG enumerations providing support for an IANA-maintained algorithm registry.

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 draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* EEEE --> 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:

* 2024-03-16 --> the publication date of this draft

The "Relation to other RFCs" section Section 1.2 contains the text "one or more YANG modules" and, later, "modules". This text is sourced from a file in a context where it is unknown how many modules a draft defines. The text is not wrong as is, but it may be improved by stating more directly how many modules are defined.

The "Relation to other RFCs" section Section 1.2 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

The following Appendix sections are to be removed prior to publication:

* Appendix A.1. Initial Module for the "Encryption Algorithm Names" Registry

* Appendix A.2. Initial Module for the "MAC Algorithm Names" Registry

* Appendix A.3. Initial Module for the "Public Key Algorithm Names" Registry

* Appendix A.4. Initial Module for the "Key Exchange Method Names" Registry

* Appendix B. Change Log

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

This document presents seven YANG 1.1 [RFC7950] modules. Three "IETF" modules and four "IANA" modules.

The three IETF modules are ietf-ssh-common (Section 2), ietf-ssh-client (Section 3), and ietf-ssh-server (Section 4). The "ietf-ssh-client" and "ietf-ssh-server" modules are the primary productions of this work, supporting the configuration and monitoring of SSH clients and servers.

The groupings defined in this document are expected to be used in conjunction with the groupings defined in an underlying transport-level module, such as the groupings defined in [I-D.ietf-netconf-tcp-client-server]. The transport-level data model enables the configuration of transport-level values such as a remote address, a remote port, a local address, and a local port.

The four IANA modules are: iana-ssh-encryption-algs (Appendix A.1), iana-ssh-key-exchange-algs (Appendix A.4), iana-ssh-mac-algs (Appendix A.2), and iana-ssh-public-key-algs (Appendix A.3). These modules each define YANG enumerations providing support for an IANA-maintained algorithm registry.

This document assumes that the four IANA modules exist, and presents a script in Appendix A that IANA may use to generate the YANG modules. This document does not publish initial versions of these four modules. IANA publishes these modules.

1.1. Regarding the IETF Modules

The three IETF modules define features and groupings to model "generic" SSH clients and SSH servers, where "generic" should be interpreted as "least common denominator" rather than "complete." Basic SSH protocol ([RFC4252], [RFC4253], and [RFC4254]) support is afforded by these modules, leaving configuration of advance features (e.g., multiple channels) to augmentations made by consuming modules.

It is intended that the YANG groupings will be used by applications needing to configure SSH client and server protocol stacks. For instance, these groupings are used to help define the data model for NETCONF over SSH [RFC6242] based clients and servers in [I-D.ietf-netconf-netconf-client-server].

The ietf-ssh-client and ietf-ssh-server YANG modules each define one grouping, which is focused on just SSH-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen on or connect to. This affords applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "ssh-server-grouping" grouping for the SSH parts it provides, while adding data nodes for the TCP-level call-home configuration.

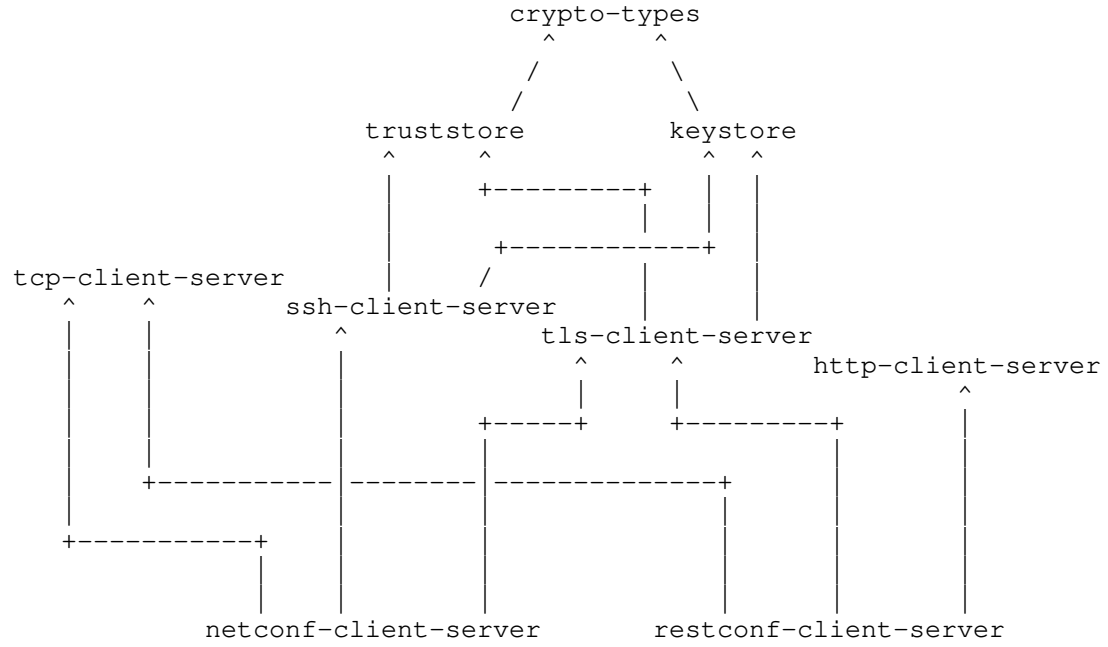
The modules defined in this document optionally support [RFC6187] enabling X.509v3 certificate based host keys and public keys.

1.2. Relation to other RFCs

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

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



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

Table 1: Label in Diagram to RFC Mapping

1.3. 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.

1.4. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational> (Section 5.3 of [RFC8342]), and <system> [I-D.ietf-netmod-system-config], if implemented.

1.5. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-ssh-common" Module

The SSH common model presented in this section contains features and groupings common to both SSH clients and SSH servers. The "transport-params-grouping" grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of permitted algorithms are in decreasing order of usage preference. The algorithm that appears first in the client list that also appears in the server list is the one that is used for the SSH transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

2.1. Data Model Overview

This section provides an overview of the "ietf-ssh-common" module in terms of its features, identities, and groupings.

2.1.1. Features

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

```
Features:
  +-- ssh-x509-certs
  +-- transport-params
  +-- asymmetric-key-pair-generation
  +-- algorithm-discovery
```

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

Please refer to the YANG module for a description of each feature.

2.1.2. Groupings

The "ietf-ssh-common" module defines the following "grouping" statement:

* transport-params-grouping

This grouping is presented in the following subsection.

2.1.2.1. The "transport-params-grouping" Grouping

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

```
  grouping transport-params-grouping:
    +-- host-key
    |  +-- host-key-alg*   ssh-public-key-algorithm
    +-- key-exchange
    |  +-- key-exchange-alg*   ssh-key-exchange-algorithm
    +-- encryption
    |  +-- encryption-alg*   ssh-encryption-algorithm
    +-- mac
       +-- mac-alg*   ssh-mac-algorithm
```

Comments:

* This grouping is used by both the "ssh-client-grouping" and the "ssh-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.

* This grouping enables client and server configurations to specify the algorithms that are to be used when establishing SSH sessions.

* Each list is "ordered-by user".

2.1.3. Protocol-accessible Nodes

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

```
module: ietf-ssh-common
  +--ro supported-algorithms {algorithm-discovery}?
     +--ro public-key-algorithms
     |  +--ro supported-algorithm*   ssh-public-key-algorithm
     +--ro encryption-algorithms
     |  +--ro supported-algorithm*   ssh-encryption-algorithm
     +--ro key-exchange-algorithms
     |  +--ro supported-algorithm*   ssh-key-exchange-algorithm
     +--ro mac-algorithms
        +--ro supported-algorithm*   ssh-mac-algorithm

  rpcs:
    +---x generate-asymmetric-key-pair
            {asymmetric-key-pair-generation}?
       +---w input
       |  +---w algorithm               ssh-public-key-algorithm
       |  +---w num-bits?               uint16
       |  +---w private-key-encoding
       |     +---w (private-key-encoding)
       |        +--:(cleartext) {ct:cleartext-private-keys}?
       |        |  +---w cleartext?   empty
       |        +--:(encrypted) {ct:encrypted-private-keys}?
       |        |  +---w encrypted
       |        |     +---w ks:encrypted-by-grouping
       |        +--:(hidden) {ct:hidden-private-keys}?
       |           +---w hidden?      empty
       +--ro output
          +--ro (key-or-hidden)?
             +--:(key)
             |  +---u ct:asymmetric-key-pair-grouping
             +--:(hidden)
                +--ro location?
                        instance-identifier
```

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-ssh-common" module are limited to "supported-algorithms" container, which is constrained by the "algorithm-discovery" feature, and the RPC "generate-asymmetric-key-pair", which is constrained by the "asymmetric-key-pair-generation" feature.

* The "encrypted-by-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-keystore].

* The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.6 of [I-D.ietf-netconf-crypto-types].

2.2. Example Usage

The following example illustrates the "transport-params-grouping' grouping when populated with some data.

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

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<transport-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
  <host-key>
    <host-key-alg>x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>ssh-rsa</host-key-alg>
    <host-key-alg>ssh-rsa@openssh.com</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>diffie-hellman-group-exchange-sha256</key-exch\
ange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>aes256-ctr</encryption-alg>
    <encryption-alg>aes192-ctr</encryption-alg>
    <encryption-alg>aes128-ctr</encryption-alg>
    <encryption-alg>aes256-gcm@openssh.com</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>hmac-sha2-256</mac-alg>
    <mac-alg>hmac-sha2-512</mac-alg>
  </mac>
</transport-params>
```

The following example illustrates operational state data indicating the SSH algorithms supported by the server.

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

   <supported-algorithms
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
     <encryption-algorithms>
       <supported-algorithm>aes256-ctr</supported-algorithm>
       <supported-algorithm>arcfour256</supported-algorithm>
       <supported-algorithm>serpent256-ctr</supported-algorithm>
       <supported-algorithm>AEAD_AES_128_GCM</supported-algorithm>
       <supported-algorithm>AEAD_AES_256_GCM</supported-algorithm>
       <supported-algorithm>aes256-gcm@openssh.com</supported-algorithm>
     </encryption-algorithms>
     <key-exchange-algorithms>
       <supported-algorithm>ecdh-sha2-nistp256</supported-algorithm>
       <supported-algorithm>rsa2048-sha256</supported-algorithm>
       <supported-algorithm>gss-group14-sha1-nistp256</supported-algori\
   thm>
       <supported-algorithm>gss-gex-sha1-nistp256</supported-algorithm>
       <supported-algorithm>gss-group14-sha256-1.2.840.10045.3.1.1</sup\
   ported-algorithm>
       <supported-algorithm>curve25519-sha256</supported-algorithm>
     </key-exchange-algorithms>
     <mac-algorithms>
       <supported-algorithm>hmac-sha2-256</supported-algorithm>
       <supported-algorithm>hmac-sha2-512</supported-algorithm>
       <supported-algorithm>AEAD_AES_256_GCM</supported-algorithm>
     </mac-algorithms>
     <public-key-algorithms>
       <supported-algorithm>rsa-sha2-256</supported-algorithm>
       <supported-algorithm>rsa-sha2-512</supported-algorithm>
       <supported-algorithm>spki-sign-rsa</supported-algorithm>
       <supported-algorithm>pgp-sign-dss</supported-algorithm>
       <supported-algorithm>x509v3-rsa2048-sha256</supported-algorithm>
       <supported-algorithm>ecdsa-sha2-nistp256</supported-algorithm>
       <supported-algorithm>ecdsa-sha2-1.3.132.0.37</supported-algorith\
   m>
       <supported-algorithm>ssh-ed25519</supported-algorithm>
       <supported-algorithm>ssh-rsa@openssh.com</supported-algorithm>
     </public-key-algorithms>
   </supported-algorithms>

   The following example illustrates the "generate-asymmetric-key-pair"
   RPC.

   REQUEST
```

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

   <rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <generate-asymmetric-key-pair
       xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
       <algorithm>ecdsa-sha2-nistp256</algorithm>
       <num-bits>521</num-bits>
       <private-key-encoding>
         <encrypted>
           <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-key-re\
   f>
         </encrypted>
       </private-key-encoding>
     </generate-asymmetric-key-pair>
   </rpc>

   RESPONSE

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

   <rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
     xmlns:sshcmn="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
     <sshcmn:public-key-format>ct:subject-public-key-info-format</sshcm\
   n:public-key-format>
     <sshcmn:public-key>BASE64VALUE=</sshcmn:public-key>
     <sshcmn:private-key-format>ct:ec-private-key-format</sshcmn:privat\
   e-key-format>
     <sshcmn:cleartext-private-key>BASE64VALUE=</sshcmn:cleartext-priva\
   te-key>
   </rpc-reply>
```

2.3. YANG Module

This YANG module has normative references to [RFC4253], [RFC4344], [RFC4419], [RFC5656], [RFC6187], [RFC6668], and [FIPS_186-6].

```
   <CODE BEGINS> file "ietf-ssh-common@2024-03-16.yang"

   module ietf-ssh-common {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
     prefix sshcmn;

     import iana-ssh-encryption-algs {
       prefix sshea;
```

```
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    import iana-ssh-key-exchange-algs {
      prefix sshkea;
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    import iana-ssh-mac-algs {
      prefix sshma;
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    import iana-ssh-public-key-algs {
      prefix sshpka;
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

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

    import ietf-keystore {
      prefix ks;
      reference
        "RFC CCCC: A YANG Data Model for a Keystore";
    }

    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 <mailto:kent+ietf@watsen.net>
       Author:   Gary Wu <mailto:garywu@cisco.com>";

    description
      "This module defines a common features and groupings for
       Secure Shell (SSH).

       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 EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); 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 2024-03-16 {
        description
          "Initial version";
        reference
          "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
      }

      // Features

      feature ssh-x509-certs {
        description
          "X.509v3 certificates are supported for SSH.";
        reference
          "RFC 6187: X.509v3 Certificates for Secure Shell
                     Authentication";
      }

      feature transport-params {
        description
          "SSH transport layer parameters are configurable.";
      }

      feature asymmetric-key-pair-generation {
        description
          "Indicates that the server implements the
           'generate-asymmetric-key-pair' RPC.";
      }
```

```
    feature algorithm-discovery {
      description
        "Indicates that the server implements the
         'supported-algorithms' container.";
    }

    // Typedefs

    typedef ssh-public-key-algorithm {
      type union {
        type sshpka:ssh-public-key-algorithm;
        type string {
          length "1..64" {
            description
              "Non IANA-maintained algorithms must include the
               'at-sign' (@) in them, per Section 4.6.1 of RFC
               4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
          pattern ".*@.*" {
            description
              "Non IANA-maintained algorithms must include the
               'at-sign' (@) in them, per Section 4.6.1 of RFC
               4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
        }
      }
      description
        "A type that enables the public key algorithm to be
         either an IANA-maintained public key algorithm in
         the 'iana-ssh-public-key-algs' YANG module (RFC EEEE),
         or a locally-defined algorithm, per Section 4.6.1
         of RFC 4250.";
      reference
        "RFC 4250: SSH Protocol Assigned Numbers
         RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    typedef ssh-key-exchange-algorithm {
      type union {
        type sshkea:ssh-key-exchange-algorithm;
        type string {
          length "1..64" {
            description
              "Non IANA-maintained algorithms must include the
```

```
              'at-sign' (@) in them, per Section 4.6.1 of RFC
              4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
          pattern ".*@.*" {
            description
              "Non IANA-maintained algorithms must include the
               'at-sign' (@) in them, per Section 4.6.1 of RFC
               4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
        }
      }
      description
        "A type that enables the key exchange algorithm to be
         either an IANA-maintained key exchange algorithm in
         the 'iana-ssh-key-exchange-algs' YANG module (RFC EEEE),
         or a locally-defined algorithm, per Section 4.6.1
         of RFC 4250.";
      reference
        "RFC 4250: SSH Protocol Assigned Numbers
         RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    typedef ssh-encryption-algorithm {
      type union {
        type sshea:ssh-encryption-algorithm;
        type string {
          length "1..64" {
            description
              "Non IANA-maintained algorithms must include the
               'at-sign' (@) in them, per Section 4.6.1 of RFC
               4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
          pattern ".*@.*" {
            description
              "Non IANA-maintained algorithms must include the
               'at-sign' (@) in them, per Section 4.6.1 of RFC
               4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
        }
      }
```

```
      description
        "A type that enables the encryption algorithm to be
         either an IANA-maintained encryption algorithm in
         the 'iana-ssh-encryption-algs' YANG module (RFC EEEE),
         or a locally-defined algorithm, per Section 4.6.1
         of RFC 4250.";
      reference
        "RFC 4250: SSH Protocol Assigned Numbers
         RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    typedef ssh-mac-algorithm {
      type union {
        type sshma:ssh-mac-algorithm;
        type string {
          length "1..64" {
            description
              "Non IANA-maintained algorithms must include the
               'at-sign' (@) in them, per Section 4.6.1 of RFC
               4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
          pattern ".*@.*" {
            description
              "Non IANA-maintained algorithms must include the
               'at-sign' (@) in them, per Section 4.6.1 of RFC
               4250.";
            reference
              "RFC 4250: SSH Protocol Assigned Numbers";
          }
        }
      }
      description
        "A type that enables the MAC algorithm to be
         either an IANA-maintained MAC algorithm in
         the 'iana-ssh-mac-algs' YANG module (RFC EEEE),
         or a locally-defined algorithm, per Section 4.6.1
         of RFC 4250.";
      reference
        "RFC 4250: SSH Protocol Assigned Numbers
         RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Groupings

    grouping transport-params-grouping {
      description
```

```
        "A reusable grouping for SSH transport parameters.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      container host-key {
        description
          "Parameters regarding host key.";
        leaf-list host-key-alg {
          type ssh-public-key-algorithm;
          ordered-by user;
          description
            "Acceptable host key algorithms in order of decreasing
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable host key algorithms are implementation-
             defined.";
          reference
            "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
        }
      }
      container key-exchange {
        description
          "Parameters regarding key exchange.";
        leaf-list key-exchange-alg {
          type ssh-key-exchange-algorithm;
          ordered-by user;
          description
            "Acceptable key exchange algorithms in order of decreasing
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable key exchange algorithms are implementation
             defined.";
        }
      }
      container encryption {
        description
          "Parameters regarding encryption.";
        leaf-list encryption-alg {
          type ssh-encryption-algorithm;
          ordered-by user;
          description
            "Acceptable encryption algorithms in order of decreasing
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable encryption algorithms are implementation
             defined.";
```

```
        }
      }
      container mac {
        description
          "Parameters regarding message authentication code (MAC).";
        leaf-list mac-alg {
          type ssh-mac-algorithm;
          ordered-by user;
          description
            "Acceptable MAC algorithms in order of decreasing
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable MAC algorithms are implementation-
             defined.";
        }
      }
    }

    // Protocol-accessible Nodes

    container supported-algorithms {
      if-feature "algorithm-discovery";
      config false;
      description
        "Identifies all of the supported algorithms.";
      container public-key-algorithms {
        description
          "A container for a list of public key algorithms
           supported by the server.";
        leaf-list supported-algorithm {
          type ssh-public-key-algorithm;
          description
            "A public key algorithm supported by the server.";
        }
      }
      container encryption-algorithms {
        description
          "A container for a list of encryption algorithms
           supported by the server.";
        leaf-list supported-algorithm {
          type ssh-encryption-algorithm;
          description
            "An encryption algorithm supported by the server.";
        }
      }
      container key-exchange-algorithms {
        config false;
```

```
        description
          "A container for a list of key exchange algorithms
           supported by the server.";
        leaf-list supported-algorithm {
          type ssh-key-exchange-algorithm;
          description
            "A key exchange algorithm supported by the server.";
        }
      }
      container mac-algorithms {
        config false;
        description
          "A container for a list of MAC algorithms
           supported by the server.";
        leaf-list supported-algorithm {
          type ssh-mac-algorithm;
          description
            "A MAC algorithm supported by the server.";
        }
      }
    }

    rpc generate-asymmetric-key-pair {
      if-feature "asymmetric-key-pair-generation";
      description
        "Requests the device to generate a public key using
         the specified key algorithm.";
      input {
        leaf algorithm {
          type ssh-public-key-algorithm;
          mandatory true;
          description
            "The algorithm to be used when generating the key.";
        }
        leaf num-bits {
          type uint16;
          description
            "Specifies the number of bits in the key to create.
             For RSA keys, the minimum size is 1024 bits and
             the default is 3072 bits. Generally, 3072 bits is
             considered sufficient. DSA keys must be exactly 1024
             bits as specified by FIPS 186-6.  For ECDSA keys, the
             'num-bits' value determines the key length by selecting
             from one of three elliptic curve sizes: 256, 384 or
             521 bits. Attempting to use bit lengths other than
             these three values for ECDSA keys will fail. ECDSA-SK,
             Ed25519 and Ed25519-SK keys have a fixed length and
             thus the 'num-bits' value is not specified.";
```

```
          reference
            "FIPS 186-6: Digital Signature Standard (DSS)";
        }
        container private-key-encoding {
          description
            "Indicates how the private key is to be encoded.";
          choice private-key-encoding {
            mandatory true;
            description
              "A choice amongst optional private key handling.";
            case cleartext {
              if-feature "ct:cleartext-private-keys";
              leaf cleartext {
                type empty;
                description
                  "Indicates that the private key is to be returned
                   as a cleartext value.";
              }
            }
            case encrypted {
              if-feature "ct:encrypted-private-keys";
              container encrypted {
                description
                   "Indicates that the private key is to be encrypted
                    using the specified symmetric or asymmetric key.";
                uses ks:encrypted-by-grouping;
              }
            }
            case hidden {
              if-feature "ct:hidden-private-keys";
              leaf hidden {
                type empty;
                description
                  "Indicates that the private key is to be hidden.

                   Unlike the 'cleartext' and 'encrypt' options, the
                   key returned is a placeholder for an internally
                   stored key.  See the 'Support for Built-in Keys'
                   section in RFC CCCC for information about hidden
                   keys.

                   It is expected that the server will instantiate
                   the hidden key in the same location where built-in
                   keys are located.  Rather than return the key,
                   just the key's location is returned in the output.";
              }
            }
          }
```

```
        }
      }
      output {
        choice key-or-hidden {
          case key {
            uses ct:asymmetric-key-pair-grouping;
          }
          case hidden {
            leaf location {
              type instance-identifier;
              description
                "The location to where a hidden key was created.";
            }
          }
          description
            "The output can be either a key (for cleartext and
             encrypted keys) or the location to where the key
             was created (for hidden keys).";
        }
      }
    } // end generate-asymmetric-key-pair

  }
```

<CODE ENDS>

3. The "ietf-ssh-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-ssh-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-ssh-client" module in terms of its features and groupings.

3.1.1. Features

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

```
Features:
  +-- ssh-client-keepalives
  +-- client-ident-password
  +-- client-ident-publickey
  +-- client-ident-hostbased
  +-- client-ident-none
```

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

Please refer to the YANG module for a description of each feature.

3.1.2. Groupings

The "ietf-ssh-client" module defines the following "grouping" statement:

* ssh-client-grouping

This grouping is presented in the following subsection.

3.1.2.1. The "ssh-client-grouping" Grouping

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

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

     grouping ssh-client-grouping:
       +-- client-identity
       |  +-- username?      string
       |  +-- public-key! {client-ident-publickey}?
       |  |  +---u ks:inline-or-keystore-asymmetric-key-grouping
       |  +-- password! {client-ident-password}?
       |  |  +---u ct:password-grouping
       |  +-- hostbased! {client-ident-hostbased}?
       |  |  +---u ks:inline-or-keystore-asymmetric-key-grouping
       |  +-- none?          empty {client-ident-none}?
       |  +-- certificate! {sshcmn:ssh-x509-certs}?
       |     +---u ks:inline-or-keystore-end-entity-cert-with-key-group\
   ing
       +-- server-authentication
       |  +-- ssh-host-keys!
       |  |  +---u ts:inline-or-truststore-public-keys-grouping
       |  +-- ca-certs! {sshcmn:ssh-x509-certs}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- ee-certs! {sshcmn:ssh-x509-certs}?
       |     +---u ts:inline-or-truststore-certs-grouping
       +-- transport-params {sshcmn:transport-params}?
       |  +---u sshcmn:transport-params-grouping
       +-- keepalives! {ssh-client-keepalives}?
          +-- max-wait?       uint16
          +-- max-attempts?   uint8
```

Comments:

* The "client-identity" node configures a "username" and authentication methods, each enabled by a "feature" statement defined in Section 3.1.1.

* The "server-authentication" node configures trust anchors for authenticating the SSH server, with each option enabled by a "feature" statement.

* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH server. The aliveness-test occurs at the SSH protocol layer.

* For the referenced grouping statement(s):

- The "inline-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
- The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
- The "inline-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-trust-anchors].
- The "inline-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-trust-anchors].
- The "transport-params-grouping" grouping is discussed in Section 2.1.2.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-ssh-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "ssh-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a inlined key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

The following configuration example uses inline-definitions for the client identity and server authentication:

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

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

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

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <username>foobar</username>
    <public-key>
      <inline-definition>
        <private-key-format>ct:rsa-private-key-format</private-key-f\
```

```
   ormat>
           <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         </inline-definition>
       </public-key>
     </client-identity>

     <!-- which host keys will this client trust -->
     <server-authentication>
       <ssh-host-keys>
         <inline-definition>
           <public-key>
             <name>corp-fw1</name>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   rmat>
             <public-key>BASE64VALUE=</public-key>
           </public-key>
           <public-key>
             <name>corp-fw2</name>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   rmat>
             <public-key>BASE64VALUE=</public-key>
           </public-key>
         </inline-definition>
       </ssh-host-keys>
       <ca-certs>
         <inline-definition>
           <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>
         </inline-definition>
       </ca-certs>
       <ee-certs>
         <inline-definition>
           <certificate>
             <name>My Application #1</name>
             <cert-data>BASE64VALUE=</cert-data>
           </certificate>
           <certificate>
             <name>My Application #2</name>
             <cert-data>BASE64VALUE=</cert-data>
           </certificate>
         </inline-definition>
       </ee-certs>
```

```
    </server-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-client>
```

The following configuration example uses central-keystore-references for the client identity and central-truststore-references for server authentication: from the keystore:

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

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <ssh-client
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
     xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

     <!-- how this client will authenticate itself to the server -->
     <client-identity>
       <username>foobar</username>
       <public-key>
         <central-keystore-reference>ssh-rsa-key</central-keystore-refe\
   rence>
       </public-key>
       <certificate>
         <central-keystore-reference>
           <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
           <certificate>ex-rsa-cert2</certificate>
         </central-keystore-reference>
       </certificate>
     </client-identity>

     <!-- which host-keys will this client trust -->
     <server-authentication>
       <ssh-host-keys>
         <central-truststore-reference>trusted-ssh-public-keys</central\
   -truststore-reference>
       </ssh-host-keys>
       <ca-certs>
         <central-truststore-reference>trusted-server-ca-certs</central\
   -truststore-reference>
       </ca-certs>
       <ee-certs>
         <central-truststore-reference>trusted-server-ee-certs</central\
   -truststore-reference>
       </ee-certs>
     </server-authentication>

     <keepalives>
       <max-wait>30</max-wait>
       <max-attempts>3</max-attempts>
     </keepalives>

   </ssh-client>
```

3.3. YANG Module

This YANG module has normative references to [RFC4252], [RFC4254], [RFC8341], [I-D.ietf-netconf-crypto-types], [I-D.ietf-netconf-trust-anchors], and [I-D.ietf-netconf-keystore].

<CODE BEGINS> file "ietf-ssh-client@2024-03-16.yang"

```
module ietf-ssh-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
  prefix sshc;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-ssh-common {
    prefix sshcmn;
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  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 <mailto:kent+ietf@watsen.net>";

      description
        "This module defines a reusable grouping for SSH clients that
         can be used as a basis for specific SSH client instances.

         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 EEEE
         (https://www.rfc-editor.org/info/rfcEEEE); 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 2024-03-16 {
        description
          "Initial version";
        reference
          "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
      }

      // Features

      feature ssh-client-keepalives {
        description
          "Per socket SSH keepalive parameters are configurable for
           SSH clients on the server implementing this feature.";
      }

      feature client-ident-publickey {
        description
          "Indicates that the 'publickey' authentication type, per
           RFC 4252, is supported for client identification.
           The 'publickey' authentication type is required by
```

```
          RFC 4252, but common implementations allow it to
          be disabled.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    feature client-ident-password {
      description
        "Indicates that the 'password' authentication type, per
         RFC 4252, is supported for client identification.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    feature client-ident-hostbased {
      description
        "Indicates that the 'hostbased' authentication type, per
         RFC 4252, is supported for client identification.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    feature client-ident-none {
      description
        "Indicates that the 'none' authentication type, per
         RFC 4252, is supported for client identification.
         It is NOT RECOMMENDED to enable this feature.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    // Groupings

    grouping ssh-client-grouping {
      description
        "A reusable grouping for configuring a SSH client without
         any consideration for how an underlying TCP session is
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a nesting of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
```

```
          'ssh-client-parameters').  This model purposely does
          not do this itself so as to provide maximum flexibility
          to consuming models.";

        container client-identity {
          nacm:default-deny-write;
          description
            "The username and authentication methods for the client.
             The authentication methods are unordered.  Clients may
             initially send any configured method or, per RFC 4252,
             Section 5.2, send the 'none' method to prompt the server
             to provide a list of productive methods.  Whenever a
             choice amongst methods arises, implementations SHOULD
             use a default ordering that prioritizes automation
             over human-interaction.";
          leaf username {
            type string;
            description
              "The username of this user.  This will be the username
               used, for instance, to log into an SSH server.";
          }
          container public-key {
            if-feature "client-ident-publickey";
            presence
              "Indicates that publickey-based authentication has been
               configured.  This statement is present so the mandatory
               descendant nodes do not imply that this node must be
               configured.";
            description
              "A locally-defined or referenced asymmetric key
               pair to be used for client identification.";
            reference
              "RFC CCCC: A YANG Data Model for a Keystore";
            uses ks:inline-or-keystore-asymmetric-key-grouping {
              refine "inline-or-keystore/inline/inline-definition" {
                must 'not(public-key-format) or derived-from-or-self'
                   + '(public-key-format, "ct:ssh-public-key-format")';
              }
              refine "inline-or-keystore/central-keystore/"
                   + "central-keystore-reference" {
                must 'not(deref(.)/../ks:public-key-format) or derived-'
                   + 'from-or-self(deref(.)/../ks:public-key-format, '
                   + '"ct:ssh-public-key-format")';
              }
            }
          }
          container password {
            if-feature "client-ident-password";
```

```
          presence
            "Indicates that password-based authentication has been
             configured.  This statement is present so the mandatory
             descendant nodes do not imply that this node must be
             configured.";
          description
            "A password to be used to authenticate the client's
             identity.";
          uses ct:password-grouping;
        }
        container hostbased {
          if-feature "client-ident-hostbased";
          presence
            "Indicates that hostbased authentication is configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A locally-defined or referenced asymmetric key
             pair to be used for host identification.";
          reference
            "RFC CCCC: A YANG Data Model for a Keystore";
          uses ks:inline-or-keystore-asymmetric-key-grouping {
            refine "inline-or-keystore/inline/inline-definition" {
              must 'not(public-key-format) or derived-from-or-self('
                 + 'public-key-format, "ct:ssh-public-key-format")';
            }
            refine "inline-or-keystore/central-keystore/"
                 + "central-keystore-reference" {
              must 'not(deref(.)/../ks:public-key-format) or derived-'
                 + 'from-or-self(deref(.)/../ks:public-key-format, '
                 + '"ct:ssh-public-key-format")';
            }
          }
        }
        leaf none {
          if-feature "client-ident-none";
          type empty;
          description
            "Indicates that 'none' algorithm is used for client
             identification.";
        }
        container certificate {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that certificate-based authentication has been
             configured.  This statement is present so the mandatory
             descendant nodes do not imply that this node must be
             configured.";
```

```
          description
            "A locally-defined or referenced certificate
             to be used for client identification.";
          reference
            "RFC CCCC: A YANG Data Model for a Keystore";
          uses
            ks:inline-or-keystore-end-entity-cert-with-key-grouping {
            refine "inline-or-keystore/inline/inline-definition" {
              must 'not(public-key-format) or derived-from-or-self('
                 + 'public-key-format, "ct:subject-public-key-info-'
                 + 'format")';
            }
            refine "inline-or-keystore/central-keystore/"
                 + "central-keystore-reference/asymmetric-key" {
              must 'not(deref(.)/../ks:public-key-format) or derived-'
                 + 'from-or-self(deref(.)/../ks:public-key-format, '
                 + '"ct:subject-public-key-info-format")';
            }
          }
        }
      } // container client-identity

      container server-authentication {
        nacm:default-deny-write;
        must 'ssh-host-keys or ca-certs or ee-certs';
        description
          "Specifies how the SSH client can authenticate SSH servers.
           Any combination of authentication methods is additive and
           unordered.";
        container ssh-host-keys {
          presence
            "Indicates that the SSH host key have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A bag of SSH host keys used by the SSH client to
             authenticate SSH server host keys.  A server host key
             is authenticated if it is an exact match to a
             configured SSH host key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-public-keys-grouping {
            refine
              "inline-or-truststore/inline/inline-definition/public"
              + "-key" {
              must 'derived-from-or-self(public-key-format,'
                 + ' "ct:ssh-public-key-format")';
            }
```

```
            refine "inline-or-truststore/central-truststore/"
                 + "central-truststore-reference" {
              must 'not(deref(.)/../ts:public-key/ts:public-key-'
                 + 'format[not(derived-from-or-self(., "ct:ssh-'
                 + 'public-key-format"))])';
            }
          }
        }
        container ca-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that the CA certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of certificate authority (CA) certificates used by
             the SSH client to authenticate SSH servers.  A server
             is authenticated if its certificate has a valid chain
             of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that the EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of end-entity certificates used by the SSH client
             to authenticate SSH servers.  A server is authenticated
             if its certificate is an exact match to a configured
             end-entity certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
      } // container server-authentication

      container transport-params {
        nacm:default-deny-write;
        if-feature "sshcmn:transport-params";
        description
          "Configurable parameters of the SSH transport layer.";
        uses sshcmn:transport-params-grouping;
      } // container transport-parameters
```

```
      container keepalives {
        nacm:default-deny-write;
        if-feature "ssh-client-keepalives";
        presence
          "Indicates that the SSH client proactively tests the
           aliveness of the remote SSH server.";
        description
          "Configures the keep-alive policy, to proactively test
           the aliveness of the SSH server.  An unresponsive SSH
           server is dropped after approximately max-wait *
           max-attempts seconds.  Per Section 4 of RFC 4254,
           the SSH client SHOULD send an SSH_MSG_GLOBAL_REQUEST
           message with a purposely nonexistent 'request name'
           value (e.g., keepalive@ietf.org) and the 'want reply'
           value set to '1'.";
        reference
          "RFC 4254: The Secure Shell (SSH) Connection Protocol";
        leaf max-wait {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default "30";
          description
            "Sets the amount of time in seconds after which if
             no data has been received from the SSH server, a
             SSH-level message will be sent to test the
             aliveness of the SSH server.";
        }
        leaf max-attempts {
          type uint8;
          default "3";
          description
            "Sets the maximum number of sequential keep-alive
             messages that can fail to obtain a response from
             the SSH server before assuming the SSH server is
             no longer alive.";
        }
      } // container keepalives
    } // grouping ssh-client-grouping

  }

  <CODE ENDS>
```

4. The "ietf-ssh-server" Module

This section defines a YANG 1.1 module called "ietf-ssh-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-ssh-server" module in terms of its features and groupings.

4.1.1. Features

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

```
Features:
  +-- ssh-server-keepalives
  +-- local-users-supported
  +-- local-user-auth-publickey {local-users-supported}?
  +-- local-user-auth-password {local-users-supported}?
  +-- local-user-auth-hostbased {local-users-supported}?
  +-- local-user-auth-none {local-users-supported}?
```

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

Please refer to the YANG module for a description of each feature.

4.1.2. Groupings

The "ietf-ssh-server" module defines the following "grouping" statement:

* ssh-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "ssh-server-grouping" Grouping

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

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

     grouping ssh-server-grouping:
       +-- server-identity
       |  +-- host-key* [name]
       |     +-- name?                string
       |     +-- (host-key-type)
       |        +--:(public-key)
       |        |  +-- public-key
       |        |     +---u ks:inline-or-keystore-asymmetric-key-groupi\
   ng
       |        +--:(certificate)
       |           +-- certificate {sshcmn:ssh-x509-certs}?
       |              +---u ks:inline-or-keystore-end-entity-cert-with-\
   key-grouping
       +-- client-authentication
       |  +-- users {local-users-supported}?
       |  |  +-- user* [name]
       |  |     +-- name?          string
       |  |     +-- public-keys! {local-user-auth-publickey}?
       |  |     |  +---u ts:inline-or-truststore-public-keys-grouping
       |  |     +-- password
       |  |     |  +-- hashed-password?   ianach:crypt-hash
       |  |     |  |       {local-user-auth-password}?
       |  |     |  +--ro last-modified?     yang:date-and-time
       |  |     +-- hostbased! {local-user-auth-hostbased}?
       |  |     |  +---u ts:inline-or-truststore-public-keys-grouping
       |  |     +-- none?          empty {local-user-auth-none}?
       |  +-- ca-certs! {sshcmn:ssh-x509-certs}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- ee-certs! {sshcmn:ssh-x509-certs}?
       |     +---u ts:inline-or-truststore-certs-grouping
       +-- transport-params {sshcmn:transport-params}?
       |  +---u sshcmn:transport-params-grouping
       +-- keepalives! {ssh-server-keepalives}?
          +-- max-wait?       uint16
          +-- max-attempts?   uint8
```

Comments:

* The "server-identity" node configures the authentication methods the server can use to identify itself to clients. The ability to use a certificate is enabled by a "feature".

* The "client-authentication" node configures trust anchors for authenticating the SSH client, with each option enabled by a "feature" statement.

* The "transport-params" node, which must be enabled by a feature, configures parameters for the SSH sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the SSH client. The aliveness-test occurs at the SSH protocol layer.

* For the referenced grouping statement(s):

  - The "inline-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "inline-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-trust-anchors].
  - The "inline-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-trust-anchors].
  - The "transport-params-grouping" grouping is discussed in Section 2.1.2.1 in this document.

4.1.3. Protocol-accessible Nodes

The "ietf-ssh-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

4.2. Example Usage

This section presents two examples showing the "ssh-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a inlined key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

The following configuration example uses inline-definitions for the server identity and client authentication:

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

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

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

     <!-- the host-key this SSH server will present -->
     <server-identity>
       <host-key>
         <name>my-pubkey-based-host-key</name>
         <public-key>
           <inline-definition>
             <private-key-format>ct:rsa-private-key-format</private-key\
   -format>
             <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
           </inline-definition>
         </public-key>
       </host-key>
       <host-key>
         <name>my-cert-based-host-key</name>
         <certificate>
           <inline-definition>
             <private-key-format>ct:rsa-private-key-format</private-key\
   -format>
             <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
             <cert-data>BASE64VALUE=</cert-data>
           </inline-definition>
         </certificate>
       </host-key>
     </server-identity>

     <!-- the client credentials this SSH server will trust -->
     <client-authentication>
       <users>
         <user>
           <name>mary</name>
           <password>
             <hashed-password>$0$example-secret</hashed-password>
           </password>
           <public-keys>
             <inline-definition>
               <public-key>
                 <name>Mary-Key-1</name>
                 <public-key-format>ct:ssh-public-key-format</public-ke\
   y-format>
```

```
                <public-key>BASE64VALUE=</public-key>
              </public-key>
              <public-key>
                <name>Mary-Key-2</name>
                <public-key-format>ct:ssh-public-key-format</public-ke\
   y-format>
                <public-key>BASE64VALUE=</public-key>
              </public-key>
            </inline-definition>
          </public-keys>
        </user>
      </users>
      <ca-certs>
        <inline-definition>
          <certificate>
            <name>Identity Cert Issuer #1</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>Identity Cert Issuer #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </inline-definition>
      </ca-certs>
      <ee-certs>
        <inline-definition>
          <certificate>
            <name>Application #1</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>Application #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </inline-definition>
      </ee-certs>
    </client-authentication>

    <keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </keepalives>

  </ssh-server>
```

The following configuration example uses central-keystore-references for the server identity and central-truststore-references for client authentication: from the keystore:

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

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <ssh-server
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server">

     <!-- the host-key this SSH server will present -->
     <server-identity>
       <host-key>
         <name>my-pubkey-based-host-key</name>
         <public-key>
           <central-keystore-reference>ssh-rsa-key</central-keystore-re\
   ference>
         </public-key>
       </host-key>
       <host-key>
         <name>my-cert-based-host-key</name>
         <certificate>
           <central-keystore-reference>
             <asymmetric-key>ssh-rsa-key-with-cert</asymmetric-key>
             <certificate>ex-rsa-cert2</certificate>
           </central-keystore-reference>
         </certificate>
       </host-key>
     </server-identity>

     <!-- the client credentials this SSH server will trust -->
     <client-authentication>
       <users>
         <user>
           <name>mary</name>
           <password>
             <hashed-password>$0$example-secret</hashed-password>
           </password>
           <public-keys>
             <central-truststore-reference>SSH Public Keys for Applicat\
   ion A</central-truststore-reference>
           </public-keys>
         </user>
       </users>
       <ca-certs>
         <central-truststore-reference>trusted-client-ca-certs</central\
   -truststore-reference>
       </ca-certs>
       <ee-certs>
         <central-truststore-reference>trusted-client-ee-certs</central\
```

```
   -truststore-reference>
       </ee-certs>
     </client-authentication>

     <keepalives>
       <max-wait>30</max-wait>
       <max-attempts>3</max-attempts>
     </keepalives>

   </ssh-server>
```

4.3. YANG Module

This YANG module has references to [RFC4251], [RFC4252], [RFC4253], [RFC4254], [RFC7317], [RFC8341], [I-D.ietf-netconf-crypto-types], [I-D.ietf-netconf-trust-anchors], and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-ssh-server@2024-03-16.yang"

module ietf-ssh-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
  prefix sshs;

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

  import iana-crypt-hash {
    prefix ianach;
    reference
      "RFC 7317: A YANG Data Model for System Management";
  }

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-ssh-common {
    prefix sshcmn;
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  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 <mailto:kent+ietf@watsen.net>";

  description
    "This module defines a reusable grouping for SSH servers that
     can be used as a basis for specific SSH server instances.

     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 EEEE
     (https://www.rfc-editor.org/info/rfcEEEE); 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 2024-03-16 {
      description
        "Initial version";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    // Features

    feature ssh-server-keepalives {
      description
        "Per socket SSH keepalive parameters are configurable for
         SSH servers on the server implementing this feature.";
    }

    feature local-users-supported {
      description
        "Indicates that the configuration for users can be
         configured herein, as opposed to in an application
         specific location.";
    }

    feature local-user-auth-publickey {
      if-feature "local-users-supported";
      description
        "Indicates that the 'publickey' authentication type,
         per RFC 4252, is supported for locally-defined users.
         The 'publickey' authentication type is required by
         RFC 4252, but common implementations allow it to
         be disabled.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }

    feature local-user-auth-password {
      if-feature "local-users-supported";
      description
        "Indicates that the 'password' authentication type,
         per RFC 4252, is supported for locally-defined users.";
      reference
        "RFC 4252:
          The Secure Shell (SSH) Authentication Protocol";
    }
```

```
     feature local-user-auth-hostbased {
       if-feature "local-users-supported";
       description
         "Indicates that the 'hostbased' authentication type,
          per RFC 4252, is supported for locally-defined users.";
       reference
         "RFC 4252:
           The Secure Shell (SSH) Authentication Protocol";
     }

     feature local-user-auth-none {
       if-feature "local-users-supported";
       description
         "Indicates that the 'none' authentication type, per
          RFC 4252, is supported.  It is NOT RECOMMENDED to
          enable this feature.";
       reference
         "RFC 4252:
           The Secure Shell (SSH) Authentication Protocol";
     }

     // Groupings

     grouping ssh-server-grouping {
       description
         "A reusable grouping for configuring a SSH server without
          any consideration for how underlying TCP sessions are
          established.

          Note that this grouping uses fairly typical descendant
          node names such that a nesting of 'uses' statements will
          have name conflicts.  It is intended that the consuming
          data model will resolve the issue (e.g., by wrapping
          the 'uses' statement in a container called
          'ssh-server-parameters').  This model purposely does
          not do this itself so as to provide maximum flexibility
          to consuming models.";

       container server-identity {
         nacm:default-deny-write;
         description
           "The list of host keys the SSH server will present when
            establishing a SSH connection.";
         list host-key {
           key "name";
           min-elements 1;
           ordered-by user;
           description
```

```
            "An ordered list of host keys (see RFC 4251) the SSH
             server will use to construct its ordered list of
             algorithms, when sending its SSH_MSG_KEXINIT message,
             as defined in Section 7.1 of RFC 4253.";
          reference
            "RFC 4251: The Secure Shell (SSH) Protocol Architecture
             RFC 4253: The Secure Shell (SSH) Transport Layer
                       Protocol";
          leaf name {
            type string;
            description
              "An arbitrary name for this host key";
          }
          choice host-key-type {
            mandatory true;
            description
              "The type of host key being specified";
            container public-key {
              description
                "A locally-defined or referenced asymmetric key pair
                 to be used for the SSH server's host key.";
              reference
                "RFC CCCC: A YANG Data Model for a Keystore";
              uses ks:inline-or-keystore-asymmetric-key-grouping {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
                   + '(public-key-format, "ct:ssh-public-key-format")';

                }
                refine "inline-or-keystore/central-keystore/"
                     + "central-keystore-reference" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                     + 'derived-from-or-self(deref(.)/../ks:public-'
                     + 'key-format, "ct:ssh-public-key-format")';
                }
              }
            }
            container certificate {
              if-feature "sshcmn:ssh-x509-certs";
              description
                "A locally-defined or referenced end-entity
                 certificate to be used for the SSH server's
                 host key.";
              reference
                "RFC CCCC: A YANG Data Model for a Keystore";
              uses
              ks:inline-or-keystore-end-entity-cert-with-key-grouping{
                refine "inline-or-keystore/inline/inline-definition" {
```

```
                must 'not(public-key-format) or derived-from-or-self'
                   + '(public-key-format, "ct:subject-public-key-'
                   + 'info-format")';
              }
              refine "inline-or-keystore/central-keystore/"
                   + "central-keystore-reference/asymmetric-key" {
                must 'not(deref(.)/../ks:public-key-format) or '
                   + 'derived-from-or-self(deref(.)/../ks:public-key'
                   + '-format, "ct:subject-public-key-info-format")';
              }
            }
          }
        }
      } // container server-identity

      container client-authentication {
        nacm:default-deny-write;
        description
          "Specifies how the SSH server can be configured to
           authenticate SSH clients.  See RFC 4252 for a general
           discussion about SSH authentication.";
        reference
          "RFC 4252: The Secure Shell (SSH) Transport Layer";
        container users {
          if-feature "local-users-supported";
          description
            "A list of locally configured users.";
          list user {
            key "name";
            description
              "A locally configured user.

               The server SHOULD derive the list of authentication
               'method names' returned to the SSH client from the
               descendant nodes configured herein, per Sections
               5.1 and 5.2 in RFC 4252.

               The authentication methods are unordered.  Clients
               must authenticate to all configured methods.
               Whenever a choice amongst methods arises,
               implementations SHOULD use a default ordering
               that prioritizes automation over human-interaction.";
            leaf name {
              type string;
              description
                "The 'user name' for the SSH client, as defined in
                 the SSH_MSG_USERAUTH_REQUEST message in RFC 4253.";
```

```
            reference
              "RFC 4253: The Secure Shell (SSH) Transport Layer
                         Protocol";
          }
          container public-keys {
            if-feature "local-user-auth-publickey";
            presence
              "Indicates that public keys have been configured.
               This statement is present so the mandatory descendant
               nodes do not imply that this node must be
               configured.";
            description
              "A set of SSH public keys may be used by the SSH
               server to authenticate this user.  A user is
               authenticated if its public key is an exact
               match to a configured public key.";
            reference
              "RFC BBBB: A YANG Data Model for a Truststore";
            uses ts:inline-or-truststore-public-keys-grouping {
              refine "inline-or-truststore/inline/inline-definition/"
                   + "public-key" {
                must 'derived-from-or-self(public-key-format,'
                   + ' "ct:ssh-public-key-format")';
              }
              refine "inline-or-truststore/central-truststore/"
                   + "central-truststore-reference" {
                must 'not(deref(.)/../ts:public-key/ts:public-key-'
                   + 'format[not(derived-from-or-self(., "ct:ssh-'
                   + 'public-key-format"))])';
              }
            }
          }
          container password {
            description
              "A password the SSH server may use to authenticate
               this user.  A user is authenticated if the hash
               of the supplied password matches this value.";
            leaf hashed-password {
              if-feature "local-user-auth-password";
              type ianach:crypt-hash;
              description
                "The password for this user.";
            }
            leaf last-modified {
              type yang:date-and-time;
              config false;
              description
                "Identifies when the password was last set.";
```

```
            }
          }
          container hostbased {
            if-feature "local-user-auth-hostbased";
            presence
              "Indicates that hostbased [RFC4252] keys have been
               configured.  This statement is present so the
               mandatory descendant nodes do not imply that this
               node must be configured.";
            description
              "A set of SSH host keys used by the SSH server to
               authenticate this user's host.  A user's host is
               authenticated if its host key is an exact match
               to a configured host key.";
            reference
              "RFC 4252: The Secure Shell (SSH) Transport Layer
               RFC BBBB: A YANG Data Model for a Truststore";
            uses ts:inline-or-truststore-public-keys-grouping {
              refine "inline-or-truststore/inline/inline-definition/"
                   + "public-key" {
                must 'derived-from-or-self(public-key-format,'
                   + ' "ct:ssh-public-key-format")';
              }
              refine "inline-or-truststore/central-truststore/"
                   + "central-truststore-reference" {
                must 'not(deref(.)/../ts:public-key/ts:public-key-'
                   + 'format[not(derived-from-or-self(., "ct:ssh-'
                   + 'public-key-format"))])';
              }
            }
          }
          leaf none {
            if-feature "local-user-auth-none";
            type empty;
            description
              "Indicates that the 'none' method is configured
               for this user.";
            reference
              "RFC 4252: The Secure Shell (SSH) Authentication
                         Protocol.";
          }
        }
      } // users
      container ca-certs {
        if-feature "sshcmn:ssh-x509-certs";
        presence
          "Indicates that CA certificates have been configured.
           This statement is present so the mandatory descendant
```

```
            nodes do not imply this node must be configured.";
          description
            "A set of certificate authority (CA) certificates used by
             the SSH server to authenticate SSH client certificates.
             A client certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "sshcmn:ssh-x509-certs";
          presence
            "Indicates that EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply this node must be configured.";
          description
            "A set of client certificates (i.e., end entity
             certificates) used by the SSH server to authenticate
             the certificates presented by SSH clients.  A client
             certificate is authenticated if it is an exact match
             to a configured end-entity certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
      } // container client-authentication

      container transport-params {
        nacm:default-deny-write;
        if-feature "sshcmn:transport-params";
        description
          "Configurable parameters of the SSH transport layer.";
        uses sshcmn:transport-params-grouping;
      } // container transport-params

      container keepalives {
        nacm:default-deny-write;
        if-feature "ssh-server-keepalives";
        presence
          "Indicates that the SSH server proactively tests the
           aliveness of the remote SSH client.";
        description
          "Configures the keep-alive policy, to proactively test
           the aliveness of the SSH client.  An unresponsive SSH
           client is dropped after approximately max-wait *
           max-attempts seconds.  Per Section 4 of RFC 4254,
           the SSH server SHOULD send an SSH_MSG_GLOBAL_REQUEST
```

```
            message with a purposely nonexistent 'request name'
            value (e.g., keepalive@ietf.org) and the 'want reply'
            value set to '1'.";
        reference
          "RFC 4254: The Secure Shell (SSH) Connection Protocol";
        leaf max-wait {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default "30";
          description
            "Sets the amount of time in seconds after which
             if no data has been received from the SSH client,
             a SSH-level message will be sent to test the
             aliveness of the SSH client.";
        }
        leaf max-attempts {
          type uint8;
          default "3";
          description
            "Sets the maximum number of sequential keep-alive
             messages that can fail to obtain a response from
             the SSH client before assuming the SSH client is
             no longer alive.";
        }
      }
    } // grouping ssh-server-grouping

  }

  <CODE ENDS>
```

5. Security Considerations

The three IETF YANG modules in this document define groupings and will not be deployed as standalone modules. Their security implications may be context dependent based on their use in other modules. The designers of modules which import these grouping must conduct their own analysis of the security considerations.

5.1. Considerations for the "iana-ssh-key-exchange-algs" Module

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

The "iana-ssh-key-exchange-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

This YANG module defines YANG enumerations, for a public IANA-maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module. IANA MAY deprecate and/or obsolete enumerations over time as needed to address security issues found in the algorithms.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. Considerations for the "iana-ssh-encryption-algs" Module

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

The "iana-ssh-encryption-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

This YANG module defines YANG enumerations, for a public IANA-maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.3. Considerations for the "iana-ssh-mac-algs" Module

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

The "iana-ssh-mac-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

This YANG module defines YANG enumerations, for a public IANA-maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.4. Considerations for the "iana-ssh-public-key-algs" Module

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

The "iana-ssh-public-key-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

This YANG module defines YANG enumerations, for a public IANA-maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.5. Considerations for the "ietf-ssh-common" YANG Module

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

The "ietf-ssh-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module defines the RPC "generate-asymmetric-key-pair" that may, if the "ct:cleartext-private-keys" feature is enabled, and the client requests it, return the private clear in cleartext form. It is NOT RECOMMENDED for private keys to pass the server's security perimeter.

This module does not define any actions or notifications, and thus the security consideration for such is not provided here.

5.6. Considerations for the "ietf-ssh-client" YANG Module

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

The "ietf-ssh-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "client-identity/password" node:

  The cleartext "password" node defined in the "ssh-client-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.7. Considerations for the "ietf-ssh-server" YANG Module

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

The "ietf-ssh-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., or even the modification of transport or keepalive parameters can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers seven URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

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

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

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

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

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

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

6.2. The "YANG Module Names" Registry

This document registers seven YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
   name:         iana-ssh-key-exchange-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs
   prefix:       sshkea
   reference:    RFC EEEE

   name:         iana-ssh-encryption-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs
   prefix:       sshea
   reference:    RFC EEEE

   name:         iana-ssh-mac-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs
   prefix:       sshma
   reference:    RFC EEEE

   name:         iana-ssh-public-key-algs
   namespace:    urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs
   prefix:       sshpka
   reference:    RFC EEEE

   name:         ietf-ssh-common
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-common
   prefix:       sshcmn
   reference:    RFC EEEE

   name:         ietf-ssh-client
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-client
   prefix:       sshc
   reference:    RFC EEEE

   name:         ietf-ssh-server
   namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-server
   prefix:       sshs
   reference:    RFC EEEE
```

6.3. Considerations for the "iana-ssh-encryption-algs" Module

This section follows the template defined in Section 4.30.3.1 of [I-D.ietf-netmod-rfc8407bis].

This document presents a script (see Appendix A) for IANA to use to generate the IANA-maintained "iana-ssh-encryption-algs" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add the following note to the registry:

> New values must not be directly added to the "iana-ssh-encryption-algs" YANG module. They must instead be added to the "Encryption Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-ENC-ALGS].

When a value is added to the "Encryption Algorithm Names" sub-registry, a new "enum" statement must be added to the "iana-ssh-encryption-algs" YANG module. The "enum" statement, and sub-statements thereof, should be defined as follows:

enum
: Replicates a name from the registry.

value
: Contains the decimal value of the IANA-assigned value.

status
: Include only if a registration has been deprecated or obsoleted. An IANA "Note" containing the word "HISTORIC" maps to YANG status "obsolete". Since the registry is unable to express a "SHOULD NOT" recommendation, there is no mapping to YANG status "deprecated".

description
: Contains "Enumeration for the 'foo-bar' algorithm.", where "foo-bar" is a placeholder for the algorithm's name (e.g., "3des-cbc").

reference
: Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-ssh-encryption-algs" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements. The "revision" must have a "description" statement explaining why the the update occurred, and must have a "reference" substatement that points to the document defining the registry update that resulted in this change. For instance:

```
   revision 2024-02-02 {
       description
         "This update reflect the update made to the underlying
          Foo Bar registry per RFC XXXX.";
     reference
         "RFC XXXX: Extend the Foo Bars Registry
                    to Support Something Important";
   }
```

IANA is requested to add the following note to the "Encryption Algorithm Names" sub-registry.

> When this registry is modified, the YANG module "iana-ssh-encryption-algs" [IANA-YANG-PARAMETERS] must be updated as defined in RFC EEEE.

6.4. Considerations for the "iana-ssh-mac-algs" Module

This section follows the template defined in Section 4.30.3.1 of [I-D.ietf-netmod-rfc8407bis].

This document presents a script (see Appendix A) for IANA to use to generate the IANA-maintained "iana-ssh-mac-algs" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add the following note to the registry:

> New values must not be directly added to the "iana-ssh-mac-algs" YANG module. They must instead be added to the "MAC Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-MAC-ALGS].

When a value is added to the "MAC Algorithm Names" sub-registry, a new "enum" statement must be added to the "iana-ssh-mac-algs" YANG module. The "enum" statement, and sub-statements thereof, should be defined as follows:

enum
: Replicates a name from the registry.

value
: Contains the decimal value of the IANA-assigned value.

status
: Include only if a registration has been deprecated or obsoleted.

description
   Contains "Enumeration for the 'foo-bar' algorithm.", where "foo-bar" is a placeholder for the algorithm's name (e.g., "3des-cbc").

reference
   Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-ssh-mac-algs" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements. The "revision" must have a "description" statement explaining why the the update occurred, and must have a "reference" substatement that points to the document defining the registry update that resulted in this change. For instance:

```
revision 2024-02-02 {
    description
      "This update reflect the update made to the underlying
       Foo Bar registry per RFC XXXX.";
  reference
      "RFC XXXX: Extend the Foo Bars Registry
                 to Support Something Important";
}
```

IANA is requested to add the following note to the "MAC Algorithm Names" sub-registry.

> When this registry is modified, the YANG module "iana-ssh-mac-algs" [IANA-YANG-PARAMETERS] must be updated as defined in RFC EEEE.

6.5. Considerations for the "iana-ssh-public-key-algs" Module

This section follows the template defined in Section 4.30.3.1 of [I-D.ietf-netmod-rfc8407bis].

This document presents a script (see Appendix A) for IANA to use to generate the IANA-maintained "iana-ssh-public-key-algs" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add the following note to the registry:

> New values must not be directly added to the "iana-ssh-public-key-algs" YANG module. They must instead be added to the "Public Key Algorithm Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-PUBKEY-ALGS].

When a value is added to the "Public Key Algorithm Names" sub-registry, a new "enum" statement must be added to the "iana-ssh-public-key-algs" YANG module. The "enum" statement, and sub-statements thereof, should be defined as follows:

enum
: Replicates a name from the registry.

value
: Contains the decimal value of the IANA-assigned value.

status
: Include only if a registration has been deprecated or obsoleted.

description
: Contains "Enumeration for the 'foo-bar' algorithm.", where "foo-bar" is a placeholder for the algorithm's name (e.g., "3des-cbc").

reference
: Replicates the reference(s) from the registry with the title of the document(s) added.

In the case that the algorithm name ends with "-*", the familiy of enumerations must be added. The familiy of enum algorithm names are generated by replacing the '*' character with these strings: "nistp256", "nistp384", "nistp521", "1.3.132.0.1", "1.2.840.10045.3.1.1", "1.3.132.0.33", "1.3.132.0.26", "1.3.132.0.27", "1.3.132.0.16", "1.3.132.0.36", "1.3.132.0.37", and "1.3.132.0.38".

Unassigned or reserved values are not present in the module.

When the "iana-ssh-public-key-algs" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements. The "revision" must have a "description" statement explaining why the the update occurred, and must have a "reference" substatement that points to the document defining the registry update that resulted in this change. For instance:

```
   revision 2024-02-02 {
       description
         "This update reflect the update made to the underlying
          Foo Bar registry per RFC XXXX.";
     reference
         "RFC XXXX: Extend the Foo Bars Registry
                    to Support Something Important";
   }
```

IANA is requested to add the following note to the "Public Key Algorithm Names" sub-registry.

> When this registry is modified, the YANG module "iana-ssh-public-key-algs" [IANA-YANG-PARAMETERS] must be updated as defined in RFC EEEE.

6.6. Considerations for the "iana-ssh-key-exchange-algs" Module

This section follows the template defined in Section 4.30.3.1 of [I-D.ietf-netmod-rfc8407bis].

This document presents a script (see Appendix A) for IANA to use to generate the IANA-maintained "iana-ssh-key-exchange-algs" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add the following note to the registry:

> New values must not be directly added to the "iana-ssh-key-exchange-algs" YANG module. They must instead be added to the "Key Exchange Method Names" sub-registry of the "Secure Shell (SSH) Protocol Parameters" registry [IANA-KEYEX-ALGS].

When a value is added to the "Key Exchange Method Names" sub-registry, a new "enum" statement must be added to the "iana-ssh-key-exchange-algs" YANG module. The "enum" statement, and sub-statements thereof, should be defined as follows:

enum
: Replicates a name from the registry.

value
: Contains the decimal value of the IANA-assigned value.

status
: Include only if a registration has been deprecated or obsoleted. An IANA "OK to Implement" containing "SHOULD NOT" maps to YANG status "deprecated". An IANA "OK to Implement" containing "MUST NOT" maps to YANG status "obsolete".

description
: Contains "Enumeration for the 'foo-bar' algorithm.", where "foo-bar" is a placeholder for the algorithm's name (e.g., "3des-cbc").

reference
: Replicates the reference(s) from the registry with the title of the document(s) added.

In the case that the algorithm name ends with "-*", the familiy of enumerations must be added. The familiy of enum algorithm names are generated by replacing the '*' character with these strings: "nistp256", "nistp384", "nistp521", "1.3.132.0.1", "1.2.840.10045.3.1.1", "1.3.132.0.33", "1.3.132.0.26", "1.3.132.0.27", "1.3.132.0.16", "1.3.132.0.36", "1.3.132.0.37", and "1.3.132.0.38".

Unassigned or reserved values are not present in the module.

When the "iana-ssh-key-exchange-algs" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements. The "revision" must have a "description" statement explaining why the the update occurred, and must have a "reference" substatement that points to the document defining the registry update that resulted in this change. For instance:

```
revision 2024-02-02 {
    description
      "This update reflect the update made to the underlying
       Foo Bar registry per RFC XXXX.";
  reference
      "RFC XXXX: Extend the Foo Bars Registry
                 to Support Something Important";
}
```

IANA is requested to add the following note to the "Key Exchange Method Names" sub-registry.

> When this registry is modified, the YANG module "iana-ssh-key-exchange-algs" [IANA-YANG-PARAMETERS] must be updated as defined in RFC EEEE.

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[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>.

[RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <https://www.rfc-editor.org/info/rfc4251>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, DOI 10.17487/RFC4254, January 2006, <https://www.rfc-editor.org/info/rfc4254>.

[RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <https://www.rfc-editor.org/info/rfc4344>.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <https://www.rfc-editor.org/info/rfc4419>.

[RFC4432] Harris, B., "RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4432, DOI 10.17487/RFC4432, March 2006, <https://www.rfc-editor.org/info/rfc4432>.

[RFC4462] Hutzelman, J., Salowey, J., Galbraith, J., and V. Welch, "Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol", RFC 4462, DOI 10.17487/RFC4462, May 2006, <https://www.rfc-editor.org/info/rfc4462>.

[RFC5647] Igoe, K. and J. Solinas, "AES Galois Counter Mode for the Secure Shell Transport Layer Protocol", RFC 5647, DOI 10.17487/RFC5647, August 2009, <https://www.rfc-editor.org/info/rfc5647>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[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>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <https://www.rfc-editor.org/info/rfc6668>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[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>.

[RFC8268] Baushke, M., "More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)", RFC 8268, DOI 10.17487/RFC8268, December 2017, <https://www.rfc-editor.org/info/rfc8268>.

[RFC8308] Bider, D., "Extension Negotiation in the Secure Shell (SSH) Protocol", RFC 8308, DOI 10.17487/RFC8308, March 2018, <https://www.rfc-editor.org/info/rfc8308>.

[RFC8332] Bider, D., "Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol", RFC 8332, DOI 10.17487/RFC8332, March 2018, <https://www.rfc-editor.org/info/rfc8332>.

[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>.

[RFC8709] Harris, B. and L. Velvindron, "Ed25519 and Ed448 Public Key Algorithms for the Secure Shell (SSH) Protocol", RFC 8709, DOI 10.17487/RFC8709, February 2020, <https://www.rfc-editor.org/info/rfc8709>.

[RFC8731] Adamantiadis, A., Josefsson, S., and M. Baushke, "Secure Shell (SSH) Key Exchange Method Using Curve25519 and Curve448", RFC 8731, DOI 10.17487/RFC8731, February 2020, <https://www.rfc-editor.org/info/rfc8731>.

[RFC8732] Sorce, S. and H. Kario, "Generic Security Service Application Program Interface (GSS-API) Key Exchange with SHA-2", RFC 8732, DOI 10.17487/RFC8732, February 2020, <https://www.rfc-editor.org/info/rfc8732>.

[RFC8758] Velvindron, L., "Deprecating RC4 in Secure Shell (SSH)", BCP 227, RFC 8758, DOI 10.17487/RFC8758, April 2020, <https://www.rfc-editor.org/info/rfc8758>.

7.2. Informative References

[FIPS_186-6]
(NIST), T. N. I. F. S. A. T., "Digital Signature Standard (DSS)", <https://csrc.nist.gov/publications/detail/fips/186/5/draft>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[I-D.ietf-netmod-rfc8407bis]
Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-09, 28 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-09>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-05, 21 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

[IANA-ENC-ALGS]
(IANA), I. A. N. A., "IANA "Encryption Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-17>.

[IANA-KEYEX-ALGS]
(IANA), I. A. N. A., "IANA "Key Exchange Method Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-16>.

[IANA-MAC-ALGS]
(IANA), I. A. N. A., "IANA "MAC Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-18>.

[IANA-PUBKEY-ALGS]
(IANA), I. A. N. A., "IANA "Public Key Algorithm Names" Sub-registry of the "Secure Shell (SSH) Protocol Parameters" Registry", <https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml#ssh-parameters-19>.

[IANA-YANG-PARAMETERS]
"YANG Parameters", n.d., <https://www.iana.org/assignments/yang-parameters>.

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

[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>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

Appendix A. Script to Generate IANA-Maintained YANG Modules

This section is not Normative.

The Python https://www.python.org script contained in this section will create the four IANA-maintained modules described in this document.

Run the script using the command `python gen-yang-modules.py`, to produce four YANG module files in the current directory.

Be aware that the script does not attempt to copy the "revision" statements from the previous/current YANG module. Copying the revision statements must be done manually.

```
<CODE BEGINS>
=============== NOTE: '\' line wrapping per RFC 8792 ================

import re
import csv
import textwrap
import requests
import requests_cache
from io import StringIO
from datetime import datetime

# Metadata for the four YANG modules produced by this script
MODULES = [
    {
        "csv_url": "https://www.iana.org/assignments/ssh-parameters/\
ssh-parameters-17.csv",
        "spaced_name": "encryption",
        "hypenated_name": "encryption",
        "prefix": "sshea",
        "description": """    "This module defines enumerations for \
the encryption algorithms
     defined in the 'Encryption Algorithm Names' sub-registry of the
     'Secure Shell (SSH) Protocol Parameters' registry maintained
     by IANA.""",
    },
    {
        "csv_url": "https://www.iana.org/assignments/ssh-parameters/\
ssh-parameters-19.csv",
        "spaced_name": "public key",
        "hypenated_name": "public-key",
        "prefix": "sshpka",
        "description": """    "This module defines enumerations for \
the public key algorithms
     defined in the 'Public Key Algorithm Names' sub-registry of the
     'Secure Shell (SSH) Protocol Parameters' registry maintained
     by IANA."""
    },
    {
        "csv_url": "https://www.iana.org/assignments/ssh-parameters/\
ssh-parameters-18.csv",
        "spaced_name": "mac",
        "hypenated_name": "mac",
        "prefix": "sshma",
        "description": """    "This module defines enumerations for \
the MAC algorithms
     defined in the 'MAC Algorithm Names' sub-registry of the
     'Secure Shell (SSH) Protocol Parameters' registry maintained
     by IANA."""
```

```
        },
        {
            "csv_url": "https://www.iana.org/assignments/ssh-parameters/\
    ssh-parameters-16.csv",
            "spaced_name": "key exchange",
            "hypenated_name": "key-exchange",
            "prefix": "sshkea",
            "description": """    "This module defines enumerations for \
    the key exchange algorithms
         defined in the 'Key Exchange Method Names' sub-registry of the
         'Secure Shell (SSH) Protocol Parameters' registry maintained
         by IANA."""
        },
    ]



    def create_module_begin(module, f):

        # Define template for all four modules
        PREAMBLE_TEMPLATE="""
    module iana-ssh-HNAME-algs {
      yang-version 1.1;
      namespace "urn:ietf:params:xml:ns:yang:iana-ssh-HNAME-algs";
      prefix PREFIX;

      organization
        "Internet Assigned Numbers Authority (IANA)";

      contact
        "Postal: ICANN
                 12025 Waterfront Drive, Suite 300
                 Los Angeles, CA  90094-2536
                 United States of America
         Tel:    +1 310 301 5800
         Email:  iana@iana.org";

      description
    DESCRIPTION

         Copyright (c) YEAR 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).

        The initial version of this YANG module is part of RFC EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
        itself for full legal notices.

        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

      revision DATE {
        description
          "This initial version of the module was created using
           the script defined in RFC EEEE to reflect the contents
           of the SNAME algorithms registry maintained by IANA.";
        reference
          "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
      }

      typedef ssh-HNAME-algorithm {
        type enumeration {
    """
        # Replacements
        rep = {
          "DATE": datetime.today().strftime('%Y-%m-%d'),
          "YEAR": datetime.today().strftime('%Y'),
          "SNAME": module["spaced_name"],
          "HNAME": module["hypenated_name"],
          "PREFIX": module["prefix"],
          "DESCRIPTION": module["description"]
        }

        # Do the replacement
        rep = dict((re.escape(k), v) for k, v in rep.items())
        pattern = re.compile("|".join(rep.keys()))
        text = pattern.sub(lambda m: rep[re.escape(m.group(0))], PREAMBL\
    E_TEMPLATE)

        # Write preamble into the file
        f.write(text)


    def create_module_body(module, f):

        # Fetch the current CSV file from IANA
        r = requests.get(module["csv_url"])
        assert(r.status_code == 200)
```

```
        # Ascertain the first CSV column's name
        with StringIO(r.text) as csv_file:
            csv_reader = csv.reader(csv_file)
            for row in csv_reader:
                first_colname = row[0]
                break

        # Parse each CSV line
        with StringIO(r.text) as csv_file:
            csv_reader = csv.DictReader(csv_file)
            for row in csv_reader:

                # Extract just the ref
                refs = row["Reference"][1:-1]  # remove the '[' and ']' \
    chars
                refs = refs.split("][")

                # There may be more than one ref
                titles = []
                for ref in refs:

                    # Ascertain the ref's title
                    if ref.startswith("RFC"):

                        # Fetch the current BIBTEX entry
                        bibtex_url="https://datatracker.ietf.org/doc/"+ \
    ref.lower() + "/bibtex/"
                        r = requests.get(bibtex_url)
                        assert r.status_code == 200, "Could not GET " + \
    bibtex_url

                        # Append to 'titles' value from the "title" line
                        for item in r.text.split("\n"):
                            if "title =" in item:
                                titles.append(re.sub('.*{{(.*)}}.*', r'\\
    g<1>', item))
                                break
                        else:
                            raise Exception("RFC title not found")


                        # Insert a space: "RFCXXXX" --> "RFC XXXX"
                        index = refs.index(ref)
                        refs[index] = "RFC " + ref[3:]

                    elif ref.startswith("FIPS"):
                        # Special case for FIPS, since no bibtex to fetch
                        if ref == "FIPS 46-3" or ref == "FIPS-46-3":
```

```
                        titles.append("Data Encryption Standard (DES\
)")
                    else:
                        raise Exception("FIPS ref not found")

                else:
                    raise Exception("ref not found")

            # Function used below
            def write_enumeration(alg):
                f.write('\n')
                f.write(f'      enum {alg} {{\n')
                if "HISTORIC" in row["Note"]:
                    f.write(f'        status obsolete;\n')
                elif "OK to Implement" in row:
                    if "MUST NOT" in row["OK to Implement"]:
                        f.write(f'        status obsolete;\n')
                    elif "SHOULD NOT" in row["OK to Implement"]:
                        f.write(f'        status deprecated;\n')
                f.write(f'        description\n')
                description = f'          "Enumeration for the \'{al\
g}\' algorithm.'
                if "Section" in row["Note"]:
                    description += " " + row["Note"]
                description += '";'
                description = textwrap.fill(description, width=69, s\
ubsequent_indent="           ")
                f.write(f'{description}\n')
                f.write('        reference\n')
                f.write('          "')
                if row["Reference"] == "":
                    f.write('    Missing in IANA registry.')
                else:
                    ref_len = len(refs)
                    for i in range(ref_len):
                        ref = refs[i]
                        f.write(f'{ref}:\n')
                        title = "             " + titles[i]
                        if i == ref_len - 1:
                            title += '";'
                        title = textwrap.fill(title, width=67, subse\
quent_indent="            ")
                        f.write(f'{title}')
                        if i != ref_len - 1:
                            f.write('\n       ')
                f.write('\n')
                f.write('      }\n')
```

```
            # Write one or more "enumeration" statements
            if not row[first_colname].endswith("-*"): # just one enu\
meration
                # Avoid duplicate entries caused by the "ecdh-sha2-*\
" family expansion
                if not row[first_colname].startswith("ecdh-sha2-nist\
p"):
                    write_enumeration(row[first_colname])
            else: # a family of enumerations
                curve_ids = [
                    "nistp256",
                    "nistp384",
                    "nistp521",
                    "1.3.132.0.1",
                    "1.2.840.10045.3.1.1",
                    "1.3.132.0.33",
                    "1.3.132.0.26",
                    "1.3.132.0.27",
                    "1.3.132.0.16",
                    "1.3.132.0.36",
                    "1.3.132.0.37",
                    "1.3.132.0.38",
                ]
                for curve_id in curve_ids:
                    write_enumeration(row[first_colname][:-1] + curv\
e_id)


def create_module_end(module, f):

    # Close out the enumeration, typedef, and module
    f.write("    }\n")
    f.write("    description\n")
    f.write(f'      "An enumeration for SSH {module["spaced_name"]} \
algorithms.";\n')
    f.write("  }\n")
    f.write('\n')
    f.write('}\n')


def create_module(module):

    # Install cache for 8x speedup
    requests_cache.install_cache()

    # ascertain yang module's name
    yang_module_name = "iana-ssh-" + module["hypenated_name"] + "-al\
gs.yang"
```

```
        # create yang module file
        with open(yang_module_name, "w") as f:
            create_module_begin(module, f)
            create_module_body(module, f)
            create_module_end(module, f)


    def main():
        for module in MODULES:
            create_module(module)


    if __name__ == "__main__":
        main()
    <CODE ENDS>
```

A.1. Initial Module for the "Encryption Algorithm Names" Registry

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2024-03-16" reflects the day on which the extraction occurred. Applications SHOULD use the IANA-maintained module, not the module defined in this draft.

This YANG module has normative references to [FIPS 46-3], [RFC4253], [RFC4344], [RFC5647], and [RFC8758].

```
<CODE BEGINS> file "iana-ssh-encryption-algs@2024-03-16.yang"

module iana-ssh-encryption-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-encryption-algs";
  prefix sshea;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
     Tel:    +1 310 301 5800
     Email:  iana@iana.org";

  description
    "This module defines enumerations for the encryption algorithms
     defined in the 'Encryption Algorithm Names' sub-registry of the
     'Secure Shell (SSH) Protocol Parameters' registry maintained
```

```
        by IANA.

        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).

        The initial version of this YANG module is part of RFC EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
        itself for full legal notices.

        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

      revision 2024-03-16 {
        description
          "This initial version of the module was created using
           the script defined in RFC EEEE to reflect the contents
           of the encryption algorithms registry maintained by IANA.";
        reference
          "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
      }

      typedef ssh-encryption-algorithm {
        type enumeration {

          enum 3des-cbc {
            description
              "Enumeration for the '3des-cbc' algorithm. Section 6.3";
            reference
              "RFC 4253:
                 The Secure Shell (SSH) Transport Layer Protocol";
          }

          enum blowfish-cbc {
            description
              "Enumeration for the 'blowfish-cbc' algorithm. Section
               6.3";
            reference
              "RFC 4253:
                 The Secure Shell (SSH) Transport Layer Protocol";
          }
```

```
        enum twofish256-cbc {
          description
            "Enumeration for the 'twofish256-cbc' algorithm. Section
             6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum twofish-cbc {
          description
            "Enumeration for the 'twofish-cbc' algorithm. Section 6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum twofish192-cbc {
          description
            "Enumeration for the 'twofish192-cbc' algorithm. Section
             6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum twofish128-cbc {
          description
            "Enumeration for the 'twofish128-cbc' algorithm. Section
             6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum aes256-cbc {
          description
            "Enumeration for the 'aes256-cbc' algorithm. Section 6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum aes192-cbc {
          description
            "Enumeration for the 'aes192-cbc' algorithm. Section 6.3";
          reference
            "RFC 4253:
```

```
            The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum aes128-cbc {
          description
            "Enumeration for the 'aes128-cbc' algorithm. Section 6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum serpent256-cbc {
          description
            "Enumeration for the 'serpent256-cbc' algorithm. Section
             6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum serpent192-cbc {
          description
            "Enumeration for the 'serpent192-cbc' algorithm. Section
             6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum serpent128-cbc {
          description
            "Enumeration for the 'serpent128-cbc' algorithm. Section
             6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum arcfour {
          status obsolete;
          description
            "Enumeration for the 'arcfour' algorithm.";
          reference
            "RFC 8758:
               Deprecating RC4 in Secure Shell (SSH)";
        }

        enum idea-cbc {
```

```
          description
            "Enumeration for the 'idea-cbc' algorithm. Section 6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum cast128-cbc {
          description
            "Enumeration for the 'cast128-cbc' algorithm. Section 6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum none {
          description
            "Enumeration for the 'none' algorithm. Section 6.3";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum des-cbc {
          status obsolete;
          description
            "Enumeration for the 'des-cbc' algorithm.";
          reference
            "FIPS-46-3:
               Data Encryption Standard (DES)";
        }

        enum arcfour128 {
          status obsolete;
          description
            "Enumeration for the 'arcfour128' algorithm.";
          reference
            "RFC 8758:
               Deprecating RC4 in Secure Shell (SSH)";
        }

        enum arcfour256 {
          status obsolete;
          description
            "Enumeration for the 'arcfour256' algorithm.";
          reference
            "RFC 8758:
               Deprecating RC4 in Secure Shell (SSH)";
```

```
        }

        enum aes128-ctr {
          description
            "Enumeration for the 'aes128-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum aes192-ctr {
          description
            "Enumeration for the 'aes192-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum aes256-ctr {
          description
            "Enumeration for the 'aes256-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum 3des-ctr {
          description
            "Enumeration for the '3des-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum blowfish-ctr {
          description
            "Enumeration for the 'blowfish-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum twofish128-ctr {
```

```
          description
            "Enumeration for the 'twofish128-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum twofish192-ctr {
          description
            "Enumeration for the 'twofish192-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum twofish256-ctr {
          description
            "Enumeration for the 'twofish256-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum serpent128-ctr {
          description
            "Enumeration for the 'serpent128-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum serpent192-ctr {
          description
            "Enumeration for the 'serpent192-ctr' algorithm.";
          reference
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
        }

        enum serpent256-ctr {
          description
            "Enumeration for the 'serpent256-ctr' algorithm.";
          reference
```

```
            "RFC 4344:
               The Secure Shell (SSH) Transport Layer Encryption
               Modes";
          }

          enum idea-ctr {
            description
              "Enumeration for the 'idea-ctr' algorithm.";
            reference
              "RFC 4344:
                 The Secure Shell (SSH) Transport Layer Encryption
                 Modes";
          }

          enum cast128-ctr {
            description
              "Enumeration for the 'cast128-ctr' algorithm.";
            reference
              "RFC 4344:
                 The Secure Shell (SSH) Transport Layer Encryption
                 Modes";
          }

          enum AEAD_AES_128_GCM {
            description
              "Enumeration for the 'AEAD_AES_128_GCM' algorithm. Section
               6.1";
            reference
              "RFC 5647:
                 AES Galois Counter Mode for the Secure Shell Transport
                 Layer Protocol";
          }

          enum AEAD_AES_256_GCM {
            description
              "Enumeration for the 'AEAD_AES_256_GCM' algorithm. Section
               6.2";
            reference
              "RFC 5647:
                 AES Galois Counter Mode for the Secure Shell Transport
                 Layer Protocol";
          }
        }
        description
          "An enumeration for SSH encryption algorithms.";
      }

    }
```

<CODE ENDS>

A.2. Initial Module for the "MAC Algorithm Names" Registry

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2024-03-16" reflects the day on which the extraction occurred. Applications SHOULD use the IANA-maintained module, not the module defined in this draft.

This YANG module has normative references [RFC4253], [RFC5647], and [RFC6668].

<CODE BEGINS> file "iana-ssh-mac-algs@2024-03-16.yang"

```
module iana-ssh-mac-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-mac-algs";
  prefix sshma;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
     Tel:    +1 310 301 5800
     Email:  iana@iana.org";

  description
    "This module defines enumerations for the MAC algorithms
     defined in the 'MAC Algorithm Names' sub-registry of the
     'Secure Shell (SSH) Protocol Parameters' registry maintained
     by IANA.

     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).

     The initial version of this YANG module is part of RFC EEEE
     (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
```

```
        itself for full legal notices.

        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

    revision 2024-03-16 {
      description
        "This initial version of the module was created using
         the script defined in RFC EEEE to reflect the contents
         of the mac algorithms registry maintained by IANA.";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    typedef ssh-mac-algorithm {
      type enumeration {

        enum hmac-sha1 {
          description
            "Enumeration for the 'hmac-sha1' algorithm. Section 6.4";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum hmac-sha1-96 {
          description
            "Enumeration for the 'hmac-sha1-96' algorithm. Section
             6.4";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum hmac-md5 {
          description
            "Enumeration for the 'hmac-md5' algorithm. Section 6.4";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum hmac-md5-96 {
          description
            "Enumeration for the 'hmac-md5-96' algorithm. Section 6.4";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
```

```
        }

        enum none {
          description
            "Enumeration for the 'none' algorithm. Section 6.4";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum AEAD_AES_128_GCM {
          description
            "Enumeration for the 'AEAD_AES_128_GCM' algorithm. Section
             6.1";
          reference
            "RFC 5647:
               AES Galois Counter Mode for the Secure Shell Transport
               Layer Protocol";
        }

        enum AEAD_AES_256_GCM {
          description
            "Enumeration for the 'AEAD_AES_256_GCM' algorithm. Section
             6.2";
          reference
            "RFC 5647:
               AES Galois Counter Mode for the Secure Shell Transport
               Layer Protocol";
        }

        enum hmac-sha2-256 {
          description
            "Enumeration for the 'hmac-sha2-256' algorithm. Section 2";
          reference
            "RFC 6668:
               SHA-2 Data Integrity Verification for the Secure Shell
               (SSH) Transport Layer Protocol";
        }

        enum hmac-sha2-512 {
          description
            "Enumeration for the 'hmac-sha2-512' algorithm. Section 2";
          reference
            "RFC 6668:
               SHA-2 Data Integrity Verification for the Secure Shell
               (SSH) Transport Layer Protocol";
        }
      }
```

```
      description
        "An enumeration for SSH mac algorithms.";
    }

  }

  <CODE ENDS>
```

A.3. Initial Module for the "Public Key Algorithm Names" Registry

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2024-03-16" reflects the day on which the extraction occurred. Applications SHOULD use the IANA-maintained module, not the module defined in this draft.

This YANG module has normative references [RFC4253], [RFC4462], [RFC5656], [RFC6187], [RFC8332], and [RFC8709].

```
<CODE BEGINS> file "iana-ssh-public-key-algs@2024-03-16.yang"

module iana-ssh-public-key-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ssh-public-key-algs";
  prefix sshpka;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
     Tel:    +1 310 301 5800
     Email:  iana@iana.org";

  description
    "This module defines enumerations for the public key algorithms
     defined in the 'Public Key Algorithm Names' sub-registry of the
     'Secure Shell (SSH) Protocol Parameters' registry maintained
     by IANA.

     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).

        The initial version of this YANG module is part of RFC EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
        itself for full legal notices.

        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

    revision 2024-03-16 {
      description
        "This initial version of the module was created using
         the script defined in RFC EEEE to reflect the contents
         of the public key algorithms registry maintained by IANA.";
      reference
        "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }

    typedef ssh-public-key-algorithm {
      type enumeration {

        enum ssh-dss {
          description
            "Enumeration for the 'ssh-dss' algorithm. Section 6.6";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum ssh-rsa {
          description
            "Enumeration for the 'ssh-rsa' algorithm. Section 6.6";
          reference
            "RFC 4253:
               The Secure Shell (SSH) Transport Layer Protocol";
        }

        enum rsa-sha2-256 {
          description
            "Enumeration for the 'rsa-sha2-256' algorithm. Section 3";
          reference
            "RFC 8332:
               Use of RSA Keys with SHA-256 and SHA-512 in the Secure
               Shell (SSH) Protocol";
        }
```

```
          enum rsa-sha2-512 {
            description
              "Enumeration for the 'rsa-sha2-512' algorithm. Section 3";
            reference
              "RFC 8332:
                 Use of RSA Keys with SHA-256 and SHA-512 in the Secure
                 Shell (SSH) Protocol";
          }

          enum spki-sign-rsa {
            description
              "Enumeration for the 'spki-sign-rsa' algorithm. Section
               6.6";
            reference
              "RFC 4253:
                 The Secure Shell (SSH) Transport Layer Protocol";
          }

          enum spki-sign-dss {
            description
              "Enumeration for the 'spki-sign-dss' algorithm. Section
               6.6";
            reference
              "RFC 4253:
                 The Secure Shell (SSH) Transport Layer Protocol";
          }

          enum pgp-sign-rsa {
            description
              "Enumeration for the 'pgp-sign-rsa' algorithm. Section
               6.6";
            reference
              "RFC 4253:
                 The Secure Shell (SSH) Transport Layer Protocol";
          }

          enum pgp-sign-dss {
            description
              "Enumeration for the 'pgp-sign-dss' algorithm. Section
               6.6";
            reference
              "RFC 4253:
                 The Secure Shell (SSH) Transport Layer Protocol";
          }

          enum null {
            description
              "Enumeration for the 'null' algorithm. Section 5";
```

```
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol";
        }

        enum ecdsa-sha2-nistp256 {
          description
            "Enumeration for the 'ecdsa-sha2-nistp256' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdsa-sha2-nistp384 {
          description
            "Enumeration for the 'ecdsa-sha2-nistp384' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdsa-sha2-nistp521 {
          description
            "Enumeration for the 'ecdsa-sha2-nistp521' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdsa-sha2-1.3.132.0.1 {
          description
            "Enumeration for the 'ecdsa-sha2-1.3.132.0.1' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdsa-sha2-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'ecdsa-sha2-1.2.840.10045.3.1.1'
             algorithm.";
          reference
```

```
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
          }

          enum ecdsa-sha2-1.3.132.0.33 {
            description
              "Enumeration for the 'ecdsa-sha2-1.3.132.0.33' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdsa-sha2-1.3.132.0.26 {
            description
              "Enumeration for the 'ecdsa-sha2-1.3.132.0.26' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdsa-sha2-1.3.132.0.27 {
            description
              "Enumeration for the 'ecdsa-sha2-1.3.132.0.27' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdsa-sha2-1.3.132.0.16 {
            description
              "Enumeration for the 'ecdsa-sha2-1.3.132.0.16' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdsa-sha2-1.3.132.0.36 {
            description
              "Enumeration for the 'ecdsa-sha2-1.3.132.0.36' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
```

```
        }

        enum ecdsa-sha2-1.3.132.0.37 {
          description
            "Enumeration for the 'ecdsa-sha2-1.3.132.0.37' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdsa-sha2-1.3.132.0.38 {
          description
            "Enumeration for the 'ecdsa-sha2-1.3.132.0.38' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum x509v3-ssh-dss {
          description
            "Enumeration for the 'x509v3-ssh-dss' algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ssh-rsa {
          description
            "Enumeration for the 'x509v3-ssh-rsa' algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-rsa2048-sha256 {
          description
            "Enumeration for the 'x509v3-rsa2048-sha256' algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-nistp256 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-nistp256'
             algorithm.";
```

```
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-nistp384 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-nistp384'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-nistp521 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-nistp521'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.1 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'x509v3-ecdsa-
             sha2-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.33 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
```

```
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.26 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.26'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.27 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.16 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.36 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.37 {
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum x509v3-ecdsa-sha2-1.3.132.0.38 {
```

```
          description
            "Enumeration for the 'x509v3-ecdsa-sha2-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 6187:
               X.509v3 Certificates for Secure Shell Authentication";
        }

        enum ssh-ed25519 {
          description
            "Enumeration for the 'ssh-ed25519' algorithm.";
          reference
            "RFC 8709:
               Ed25519 and Ed448 Public Key Algorithms for the Secure
               Shell (SSH) Protocol";
        }

        enum ssh-ed448 {
          description
            "Enumeration for the 'ssh-ed448' algorithm.";
          reference
            "RFC 8709:
               Ed25519 and Ed448 Public Key Algorithms for the Secure
               Shell (SSH) Protocol";
        }
      }
      description
        "An enumeration for SSH public key algorithms.";
    }

  }

  <CODE ENDS>
```

A.4. Initial Module for the "Key Exchange Method Names" Registry

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2024-03-16" reflects the day on which the extraction occurred. Applications SHOULD use the IANA-maintained module, not the module defined in this draft.

This YANG module has normative references to [RFC4419], [RFC4432], [RFC5656], [RFC8268], [RFC8308], [RFC8731], [RFC8732].

<CODE BEGINS> file "iana-ssh-key-exchange-algs@2024-03-16.yang"

```
   module iana-ssh-key-exchange-algs {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:iana-ssh-key-exchange-algs";
     prefix sshkea;

     organization
       "Internet Assigned Numbers Authority (IANA)";

     contact
       "Postal: ICANN
                12025 Waterfront Drive, Suite 300
                Los Angeles, CA  90094-2536
                United States of America
        Tel:    +1 310 301 5800
        Email:  iana@iana.org";

     description
       "This module defines enumerations for the key exchange algorithms
        defined in the 'Key Exchange Method Names' sub-registry of the
        'Secure Shell (SSH) Protocol Parameters' registry maintained
        by IANA.

        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).

        The initial version of this YANG module is part of RFC EEEE
        (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
        itself for full legal notices.

        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

     revision 2024-03-16 {
       description
         "This initial version of the module was created using
          the script defined in RFC EEEE to reflect the contents
          of the key exchange algorithms registry maintained by IANA.";
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }
```

```
     typedef ssh-key-exchange-algorithm {
       type enumeration {

         enum diffie-hellman-group-exchange-sha1 {
           status deprecated;
           description
             "Enumeration for the 'diffie-hellman-group-exchange-sha1'
              algorithm. Section 4.1";
           reference
             "RFC 4419:
                Diffie-Hellman Group Exchange for the Secure Shell
                (SSH) Transport Layer Protocol
          RFC 8270:
                Increase the Secure Shell Minimum Recommended Diffie-
                Hellman Modulus Size to 2048 Bits";
         }

         enum diffie-hellman-group-exchange-sha256 {
           description
             "Enumeration for the 'diffie-hellman-group-exchange-sha256'
              algorithm. Section 4.2";
           reference
             "RFC 4419:
                Diffie-Hellman Group Exchange for the Secure Shell
                (SSH) Transport Layer Protocol
          RFC 8270:
                Increase the Secure Shell Minimum Recommended Diffie-
                Hellman Modulus Size to 2048 Bits";
         }

         enum diffie-hellman-group1-sha1 {
           status deprecated;
           description
             "Enumeration for the 'diffie-hellman-group1-sha1'
              algorithm. Section 8.1";
           reference
             "RFC 4253:
                The Secure Shell (SSH) Transport Layer Protocol";
         }

         enum diffie-hellman-group14-sha1 {
           description
             "Enumeration for the 'diffie-hellman-group14-sha1'
              algorithm. Section 8.2";
           reference
             "RFC 4253:
                The Secure Shell (SSH) Transport Layer Protocol";
         }
```

```
          enum diffie-hellman-group14-sha256 {
            description
              "Enumeration for the 'diffie-hellman-group14-sha256'
               algorithm.";
            reference
              "RFC 8268:
                 More Modular Exponentiation (MODP) Diffie-Hellman (DH)
                 Key Exchange (KEX) Groups for Secure Shell (SSH)";
          }

          enum diffie-hellman-group15-sha512 {
            description
              "Enumeration for the 'diffie-hellman-group15-sha512'
               algorithm.";
            reference
              "RFC 8268:
                 More Modular Exponentiation (MODP) Diffie-Hellman (DH)
                 Key Exchange (KEX) Groups for Secure Shell (SSH)";
          }

          enum diffie-hellman-group16-sha512 {
            description
              "Enumeration for the 'diffie-hellman-group16-sha512'
               algorithm.";
            reference
              "RFC 8268:
                 More Modular Exponentiation (MODP) Diffie-Hellman (DH)
                 Key Exchange (KEX) Groups for Secure Shell (SSH)";
          }

          enum diffie-hellman-group17-sha512 {
            description
              "Enumeration for the 'diffie-hellman-group17-sha512'
               algorithm.";
            reference
              "RFC 8268:
                 More Modular Exponentiation (MODP) Diffie-Hellman (DH)
                 Key Exchange (KEX) Groups for Secure Shell (SSH)";
          }

          enum diffie-hellman-group18-sha512 {
            description
              "Enumeration for the 'diffie-hellman-group18-sha512'
               algorithm.";
            reference
              "RFC 8268:
                 More Modular Exponentiation (MODP) Diffie-Hellman (DH)
                 Key Exchange (KEX) Groups for Secure Shell (SSH)";
```

```
        }

        enum ecdh-sha2-nistp256 {
          description
            "Enumeration for the 'ecdh-sha2-nistp256' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdh-sha2-nistp384 {
          description
            "Enumeration for the 'ecdh-sha2-nistp384' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdh-sha2-nistp521 {
          description
            "Enumeration for the 'ecdh-sha2-nistp521' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdh-sha2-1.3.132.0.1 {
          description
            "Enumeration for the 'ecdh-sha2-1.3.132.0.1' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdh-sha2-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'ecdh-sha2-1.2.840.10045.3.1.1'
             algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }
```

```
          enum ecdh-sha2-1.3.132.0.33 {
            description
              "Enumeration for the 'ecdh-sha2-1.3.132.0.33' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdh-sha2-1.3.132.0.26 {
            description
              "Enumeration for the 'ecdh-sha2-1.3.132.0.26' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdh-sha2-1.3.132.0.27 {
            description
              "Enumeration for the 'ecdh-sha2-1.3.132.0.27' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdh-sha2-1.3.132.0.16 {
            description
              "Enumeration for the 'ecdh-sha2-1.3.132.0.16' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdh-sha2-1.3.132.0.36 {
            description
              "Enumeration for the 'ecdh-sha2-1.3.132.0.36' algorithm.";
            reference
              "RFC 5656:
                 Elliptic Curve Algorithm Integration in the Secure
                 Shell Transport Layer";
          }

          enum ecdh-sha2-1.3.132.0.37 {
            description
              "Enumeration for the 'ecdh-sha2-1.3.132.0.37' algorithm.";
```

```
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecdh-sha2-1.3.132.0.38 {
          description
            "Enumeration for the 'ecdh-sha2-1.3.132.0.38' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum ecmqv-sha2 {
          description
            "Enumeration for the 'ecmqv-sha2' algorithm.";
          reference
            "RFC 5656:
               Elliptic Curve Algorithm Integration in the Secure
               Shell Transport Layer";
        }

        enum gss-group1-sha1-nistp256 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-nistp256'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group1-sha1-nistp384 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-nistp384'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
```

```
            Secure Shell (SSH) Protocol
          RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group1-sha1-nistp521 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-nistp521'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group1-sha1-1.3.132.0.1 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group1-sha1-1.2.840.10045.3.1.1 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-1.2.840.10045.3.1.1'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
```

```
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group1-sha1-1.3.132.0.33 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group1-sha1-1.3.132.0.26 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-1.3.132.0.26'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group1-sha1-1.3.132.0.27 {
          status deprecated;
          description
            "Enumeration for the 'gss-group1-sha1-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }
```

```
          enum gss-group1-sha1-1.3.132.0.16 {
            status deprecated;
            description
              "Enumeration for the 'gss-group1-sha1-1.3.132.0.16'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
           RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-group1-sha1-1.3.132.0.36 {
            status deprecated;
            description
              "Enumeration for the 'gss-group1-sha1-1.3.132.0.36'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
           RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-group1-sha1-1.3.132.0.37 {
            status deprecated;
            description
              "Enumeration for the 'gss-group1-sha1-1.3.132.0.37'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
           RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-group1-sha1-1.3.132.0.38 {
            status deprecated;
            description
```

```
            "Enumeration for the 'gss-group1-sha1-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
           RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-nistp256 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-nistp256'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
           RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-nistp384 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-nistp384'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
           RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-nistp521 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-nistp521'
             algorithm.";
          reference
```

```
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
             RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-group14-sha1-1.3.132.0.1 {
            status deprecated;
            description
              "Enumeration for the 'gss-group14-sha1-1.3.132.0.1'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
             RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-group14-sha1-1.2.840.10045.3.1.1 {
            status deprecated;
            description
              "Enumeration for the 'gss-group14-sha1-1.2.840.10045.3.1.1'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
             RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-group14-sha1-1.3.132.0.33 {
            status deprecated;
            description
              "Enumeration for the 'gss-group14-sha1-1.3.132.0.33'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
```

```
            Secure Shell (SSH) Protocol
          RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-1.3.132.0.26 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-1.3.132.0.26'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-1.3.132.0.27 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-1.3.132.0.16 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
```

```
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-1.3.132.0.36 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-1.3.132.0.37 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha1-1.3.132.0.38 {
          status deprecated;
          description
            "Enumeration for the 'gss-group14-sha1-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }
```

```
          enum gss-gex-sha1-nistp256 {
            status deprecated;
            description
              "Enumeration for the 'gss-gex-sha1-nistp256' algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
           RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-gex-sha1-nistp384 {
            status deprecated;
            description
              "Enumeration for the 'gss-gex-sha1-nistp384' algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
           RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-gex-sha1-nistp521 {
            status deprecated;
            description
              "Enumeration for the 'gss-gex-sha1-nistp521' algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
           RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-gex-sha1-1.3.132.0.1 {
            status deprecated;
            description
              "Enumeration for the 'gss-gex-sha1-1.3.132.0.1'
               algorithm.";
            reference
```

```
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
             RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-gex-sha1-1.2.840.10045.3.1.1 {
            status deprecated;
            description
              "Enumeration for the 'gss-gex-sha1-1.2.840.10045.3.1.1'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
             RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-gex-sha1-1.3.132.0.33 {
            status deprecated;
            description
              "Enumeration for the 'gss-gex-sha1-1.3.132.0.33'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
                 Secure Shell (SSH) Protocol
             RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-gex-sha1-1.3.132.0.26 {
            status deprecated;
            description
              "Enumeration for the 'gss-gex-sha1-1.3.132.0.26'
               algorithm.";
            reference
              "RFC 4462:
                 Generic Security Service Application Program Interface
                 (GSS-API) Authentication and Key Exchange for the
```

```
            Secure Shell (SSH) Protocol
          RFC 8732:
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-gex-sha1-1.3.132.0.27 {
          status deprecated;
          description
            "Enumeration for the 'gss-gex-sha1-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-gex-sha1-1.3.132.0.16 {
          status deprecated;
          description
            "Enumeration for the 'gss-gex-sha1-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-gex-sha1-1.3.132.0.36 {
          status deprecated;
          description
            "Enumeration for the 'gss-gex-sha1-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
          RFC 8732:
               Generic Security Service Application Program Interface
```

```
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-gex-sha1-1.3.132.0.37 {
          status deprecated;
          description
            "Enumeration for the 'gss-gex-sha1-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-gex-sha1-1.3.132.0.38 {
          status deprecated;
          description
            "Enumeration for the 'gss-gex-sha1-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol
         RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss- {
          description
            "Enumeration for the 'gss-' algorithm. Section 2.6";
          reference
            "RFC 4462:
               Generic Security Service Application Program Interface
               (GSS-API) Authentication and Key Exchange for the
               Secure Shell (SSH) Protocol";
        }

        enum rsa1024-sha1 {
          status obsolete;
          description
            "Enumeration for the 'rsa1024-sha1' algorithm.";
          reference
```

```
            "RFC 4432:
               RSA Key Exchange for the Secure Shell (SSH) Transport
               Layer Protocol";
          }

          enum rsa2048-sha256 {
            description
              "Enumeration for the 'rsa2048-sha256' algorithm.";
            reference
              "RFC 4432:
                 RSA Key Exchange for the Secure Shell (SSH) Transport
                 Layer Protocol";
          }

          enum ext-info-s {
            description
              "Enumeration for the 'ext-info-s' algorithm. Section 2";
            reference
              "RFC 8308:
                 Extension Negotiation in the Secure Shell (SSH)
                 Protocol";
          }

          enum ext-info-c {
            description
              "Enumeration for the 'ext-info-c' algorithm. Section 2";
            reference
              "RFC 8308:
                 Extension Negotiation in the Secure Shell (SSH)
                 Protocol";
          }

          enum gss-group14-sha256-nistp256 {
            description
              "Enumeration for the 'gss-group14-sha256-nistp256'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-group14-sha256-nistp384 {
            description
              "Enumeration for the 'gss-group14-sha256-nistp384'
               algorithm.";
            reference
              "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-nistp521 {
          description
            "Enumeration for the 'gss-group14-sha256-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'gss-
             group14-sha256-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.33 {
          description
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.26 {
          description
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.26'
             algorithm.";
```

```
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.27 {
          description
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.16 {
          description
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group14-sha256-1.3.132.0.38 {
          description
```

```
            "Enumeration for the 'gss-group14-sha256-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-nistp256 {
          description
            "Enumeration for the 'gss-group15-sha512-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-nistp384 {
          description
            "Enumeration for the 'gss-group15-sha512-nistp384'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-nistp521 {
          description
            "Enumeration for the 'gss-group15-sha512-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }
```

```
        enum gss-group15-sha512-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'gss-
             group15-sha512-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-1.3.132.0.33 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-1.3.132.0.26 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.26'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-1.3.132.0.27 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-1.3.132.0.16 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
```

```
        }

        enum gss-group15-sha512-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group15-sha512-1.3.132.0.38 {
          description
            "Enumeration for the 'gss-group15-sha512-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-nistp256 {
          description
            "Enumeration for the 'gss-group16-sha512-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-nistp384 {
          description
            "Enumeration for the 'gss-group16-sha512-nistp384'
             algorithm.";
          reference
            "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-nistp521 {
          description
            "Enumeration for the 'gss-group16-sha512-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'gss-
             group16-sha512-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.33 {
          description
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.26 {
          description
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.26'
             algorithm.";
```

```
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.27 {
          description
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.16 {
          description
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group16-sha512-1.3.132.0.38 {
          description
```

```
            "Enumeration for the 'gss-group16-sha512-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-nistp256 {
          description
            "Enumeration for the 'gss-group17-sha512-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-nistp384 {
          description
            "Enumeration for the 'gss-group17-sha512-nistp384'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-nistp521 {
          description
            "Enumeration for the 'gss-group17-sha512-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }
```

```
        enum gss-group17-sha512-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'gss-
             group17-sha512-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-1.3.132.0.33 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-1.3.132.0.26 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.26'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-1.3.132.0.27 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-1.3.132.0.16 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
```

```
        }

        enum gss-group17-sha512-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group17-sha512-1.3.132.0.38 {
          description
            "Enumeration for the 'gss-group17-sha512-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-nistp256 {
          description
            "Enumeration for the 'gss-group18-sha512-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-nistp384 {
          description
            "Enumeration for the 'gss-group18-sha512-nistp384'
             algorithm.";
          reference
            "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-nistp521 {
          description
            "Enumeration for the 'gss-group18-sha512-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'gss-
             group18-sha512-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.33 {
          description
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.26 {
          description
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.26'
             algorithm.";
```

```
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.27 {
          description
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.16 {
          description
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-group18-sha512-1.3.132.0.38 {
          description
```

```
            "Enumeration for the 'gss-group18-sha512-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp256-sha256-nistp256 {
          description
            "Enumeration for the 'gss-nistp256-sha256-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp256-sha256-nistp384 {
          description
            "Enumeration for the 'gss-nistp256-sha256-nistp384'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp256-sha256-nistp521 {
          description
            "Enumeration for the 'gss-nistp256-sha256-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp256-sha256-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }
```

```
          enum gss-nistp256-sha256-1.2.840.10045.3.1.1 {
            description
              "Enumeration for the 'gss-
               nistp256-sha256-1.2.840.10045.3.1.1' algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp256-sha256-1.3.132.0.33 {
            description
              "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.33'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp256-sha256-1.3.132.0.26 {
            description
              "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.26'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp256-sha256-1.3.132.0.27 {
            description
              "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.27'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp256-sha256-1.3.132.0.16 {
            description
              "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.16'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
```

```
        }

        enum gss-nistp256-sha256-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp256-sha256-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp256-sha256-1.3.132.0.38 {
          description
            "Enumeration for the 'gss-nistp256-sha256-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-nistp256 {
          description
            "Enumeration for the 'gss-nistp384-sha384-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-nistp384 {
          description
            "Enumeration for the 'gss-nistp384-sha384-nistp384'
             algorithm.";
          reference
            "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-nistp521 {
          description
            "Enumeration for the 'gss-nistp384-sha384-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'gss-
             nistp384-sha384-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.33 {
          description
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.26 {
          description
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.26'
             algorithm.";
```

```
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.27 {
          description
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.16 {
          description
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp384-sha384-1.3.132.0.38 {
          description
```

```
            "Enumeration for the 'gss-nistp384-sha384-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp521-sha512-nistp256 {
          description
            "Enumeration for the 'gss-nistp521-sha512-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp521-sha512-nistp384 {
          description
            "Enumeration for the 'gss-nistp521-sha512-nistp384'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp521-sha512-nistp521 {
          description
            "Enumeration for the 'gss-nistp521-sha512-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp521-sha512-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }
```

```
          enum gss-nistp521-sha512-1.2.840.10045.3.1.1 {
            description
              "Enumeration for the 'gss-
               nistp521-sha512-1.2.840.10045.3.1.1' algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp521-sha512-1.3.132.0.33 {
            description
              "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.33'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp521-sha512-1.3.132.0.26 {
            description
              "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.26'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp521-sha512-1.3.132.0.27 {
            description
              "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.27'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-nistp521-sha512-1.3.132.0.16 {
            description
              "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.16'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
```

```
        }

        enum gss-nistp521-sha512-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp521-sha512-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-nistp521-sha512-1.3.132.0.38 {
          description
            "Enumeration for the 'gss-nistp521-sha512-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-nistp256 {
          description
            "Enumeration for the 'gss-curve25519-sha256-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-nistp384 {
          description
            "Enumeration for the 'gss-curve25519-sha256-nistp384'
             algorithm.";
          reference
            "RFC 8732:
```

```
            Generic Security Service Application Program Interface
            (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-nistp521 {
          description
            "Enumeration for the 'gss-curve25519-sha256-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.2.840.10045.3.1.1 {
          description
            "Enumeration for the 'gss-
             curve25519-sha256-1.2.840.10045.3.1.1' algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.33 {
          description
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.33'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.26 {
          description
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.26'
             algorithm.";
```

```
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.27 {
          description
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.27'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.16 {
          description
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.16'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve25519-sha256-1.3.132.0.38 {
          description
```

```
            "Enumeration for the 'gss-curve25519-sha256-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve448-sha512-nistp256 {
          description
            "Enumeration for the 'gss-curve448-sha512-nistp256'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve448-sha512-nistp384 {
          description
            "Enumeration for the 'gss-curve448-sha512-nistp384'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve448-sha512-nistp521 {
          description
            "Enumeration for the 'gss-curve448-sha512-nistp521'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve448-sha512-1.3.132.0.1 {
          description
            "Enumeration for the 'gss-curve448-sha512-1.3.132.0.1'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }
```

```
          enum gss-curve448-sha512-1.2.840.10045.3.1.1 {
            description
              "Enumeration for the 'gss-
               curve448-sha512-1.2.840.10045.3.1.1' algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-curve448-sha512-1.3.132.0.33 {
            description
              "Enumeration for the 'gss-curve448-sha512-1.3.132.0.33'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-curve448-sha512-1.3.132.0.26 {
            description
              "Enumeration for the 'gss-curve448-sha512-1.3.132.0.26'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-curve448-sha512-1.3.132.0.27 {
            description
              "Enumeration for the 'gss-curve448-sha512-1.3.132.0.27'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
          }

          enum gss-curve448-sha512-1.3.132.0.16 {
            description
              "Enumeration for the 'gss-curve448-sha512-1.3.132.0.16'
               algorithm.";
            reference
              "RFC 8732:
                 Generic Security Service Application Program Interface
                 (GSS-API) Key Exchange with SHA-2";
```

```
        }

        enum gss-curve448-sha512-1.3.132.0.36 {
          description
            "Enumeration for the 'gss-curve448-sha512-1.3.132.0.36'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve448-sha512-1.3.132.0.37 {
          description
            "Enumeration for the 'gss-curve448-sha512-1.3.132.0.37'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum gss-curve448-sha512-1.3.132.0.38 {
          description
            "Enumeration for the 'gss-curve448-sha512-1.3.132.0.38'
             algorithm.";
          reference
            "RFC 8732:
               Generic Security Service Application Program Interface
               (GSS-API) Key Exchange with SHA-2";
        }

        enum curve25519-sha256 {
          description
            "Enumeration for the 'curve25519-sha256' algorithm.";
          reference
            "RFC 8731:
               Secure Shell (SSH) Key Exchange Method Using
               Curve25519 and Curve448";
        }

        enum curve448-sha512 {
          description
            "Enumeration for the 'curve448-sha512' algorithm.";
          reference
            "RFC 8731:
               Secure Shell (SSH) Key Exchange Method Using
               Curve25519 and Curve448";
```

```
          }
        }
        description
          "An enumeration for SSH key exchange algorithms.";
      }

    }

    <CODE ENDS>
```

Appendix B. Change Log

B.1. 00 to 01

* Noted that '0.0.0.0' and '::' might have special meanings.

* Renamed "keychain" to "keystore".

B.2. 01 to 02

* Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.

* Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.

* Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

B.3. 02 to 03

* Removed 'RESTRICTED' enum from 'password' leaf type.

* Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

* Fixed description statement for leaf 'trusted-ca-certs'.

B.4. 03 to 04

* Change title to "YANG Groupings for SSH Clients and SSH Servers"

* Added reference to RFC 6668

* Added RFC 8174 to Requirements Language Section.

* Enhanced description statement for ietf-ssh-server's "trusted-ca-certs" leaf.

* Added mandatory true to ietf-ssh-client's "client-auth" 'choice' statement.

* Changed the YANG prefix for module ietf-ssh-common from 'sshcom' to 'sshcmn'.

* Removed the compression algorithms as they are not commonly configurable in vendors' implementations.

* Updating descriptions in transport-params-grouping and the servers's usage of it.

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated YANG to use typedefs around leafrefs to common keystore paths

* Now inlines key and certificates (no longer a leafref to keystore)

B.5. 04 to 05

* Merged changes from co-author.

B.6. 05 to 06

* Updated to use trust anchors from trust-anchors draft (was keystore draft)

* Now uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

B.7. 06 to 07

* factored the ssh-[client|server]-groupings into more reusable groupings.

* added if-feature statements for the new "ssh-host-keys" and "x509-certificates" features defined in draft-ietf-netconf-trust-anchors.

B.8. 07 to 08

* Added a number of compatibility matrices to Section 5 (thanks Frank!)

* Clarified that any configured "host-key-alg" values need to be compatible with the configured private key.

B.9. 08 to 09

* Updated examples to reflect update to groupings defined in the keystore -09 draft.

* Add SSH keepalives features and groupings.

* Prefixed top-level SSH grouping nodes with 'ssh-' and support mashups.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

* Reformatted the YANG modules.

B.11. 10 to 11

* Reformatted lines causing folding to occur.

B.12. 11 to 12

* Collapsed all the inner groupings into the top-level grouping.

* Added a top-level "demux container" inside the top-level grouping.

* Added NACM statements and updated the Security Considerations section.

* Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

B.13. 12 to 13

* Removed the "demux containers", floating the nacm:default-deny-write to each descendant node, and adding a note to model designers regarding the potential need to add their own demux containers.

* Fixed a couple references (section 2 --> section 3)

* In the server model, replaced <client-cert-auth> with <client-authentication> and introduced 'inline-or-external' choice.

B.14. 13 to 14

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

B.15. 14 to 15

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:host-keys-ref" or "ts:certificates-ref" to a container that uses "ts:inline-or-truststore-host-keys-grouping" or "ts:inline-or-truststore-certs-grouping".

B.16. 15 to 16

* Removed unnecessary if-feature statements in the -client and -server modules.

* Cleaned up some description statements in the -client and -server modules.

* Fixed a canonical ordering issue in ietf-ssh-common detected by new pyang.

B.17. 16 to 17

* Removed choice inline-or-external by removing the 'external' case and flattening the 'local' case and adding a "local-users-supported" feature.

* Updated examples to include the "*-key-format" nodes.

* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:ssh-public-key-format" (must expr for ref'ed keys are TBD).

B.18. 17 to 18

* Removed leaf-list 'other' from ietf-ssh-server.

* Removed unused 'external-client-auth-supported' feature.

* Added features client-auth-password, client-auth-hostbased, and client-auth-none.

* Renamed 'host-key' to 'public-key' for when refering to 'publickey' based auth.

* Added new feature-protected 'hostbased' and 'none' to the 'user' node's config.

* Added new feature-protected 'hostbased' and 'none' to the 'client-identity' node's config.

* Updated examples to reflect new "bag" addition to truststore.

* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.

* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags.

* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

B.19. 18 to 19

* Updated the "keepalives" containers to address Michal Vasko's request to align with RFC 8071.

* Removed algorithm-mapping tables from the "SSH Common Model" section

* Removed 'algorithm' node from examples.

* Added feature "userauth-publickey"

* Removed "choice auth-type", as auth-types are not exclusive.

* Renamed both "client-certs" and "server-certs" to "ee-certs"

* Switch "must" to assert the public-key-format is "subject-public-key-info-format" when certificates are used.

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

B.20. 19 to 20

* Added a "must 'public-key or password or hostbased or none or certificate'" statement to the "user" node in ietf-ssh-client

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

* Moved the "ietf-ssh-common" module section to proceed the other two module sections.

* Updated the Security Considerations section.

B.21. 20 to 21

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

B.22. 21 to 22

* Cleaned up the SSH-client examples (i.e., removing FIXMEs)

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

* Updated the "ietf-ssh-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

B.23. 22 to 23

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

B.24. 23 to 24

* Removed the 'supported-authentication-methods' from {grouping ssh-server-grouping}/client-authentication.

* Added XML-comment above examples explaining the reason for the unexepected top-most element's presence.

* Added RFC-references to various 'feature' statements.

* Renamed "credentials" to "authentication methods"

* Renamed "client-auth-*" to "userauth-*"

* Renamed "client-identity-*" to "userauth-*"

* Fixed nits found by YANG Doctor reviews.

* Aligned modules with 'pyang -f' formatting.

* Added a 'Contributors' section.

B.25. 24 to 25

* Moved algorithms in ietf-ssh-common (plus more) to IANA-maintained modules
* Added "config false" lists for algorithms supported by the server.
* Renamed "{ietf-ssh-client}userauth-*" to "client-ident-*"
* Renamed "{ietf-ssh-server}userauth-*" to "local-user-auth-*"
* Fixed issues found during YANG Doctor review.
* Fixed issues found during Secdir review.

B.26. 25 to 26

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
* Minor editorial nits

B.27. 26 to 27

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)
* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)
* Created identityref-based typedefs for each of the four IANA alg identity bases.
* Added ietf-ssh-common:generate-asymmetric-key-pair() RPC for discussion.

B.28. 27 to 28

* Fixed example to not have line-returns around "identity" values.
* Fixed examples to not include "xmlns:algs".
* Added an example for the "generate-asymmetric-key-pair" RPC.

B.29. 28 to 29

* Updated modules to IANA-maintained modules in Appendix A to 2022-06-16.

B.30. 29 to 30

* Fixed 'must' expressions.

* Added missing 'revision' statement.

B.31. 30 to 31

* Updated per Shepherd reviews impacting the suite of drafts.

B.32. 31 to 32

* Updated per Shepherd reviews impacting the suite of drafts.

B.33. 32 to 33

* Updated per Tom Petch review.

* Updated Intro to clarify what "generic" means.

* Added RPC-reply for 'generate-asymmetric-key-pair' example.

* Added references to RFC 4251 and FIPS 186-6.

* Added "if-feature ct:encrypted-private-keys" for "case cleartext".

B.34. 33 to 34

* Addresses AD review comments.

* Added note to Editor to fix line foldings.

* Introduction now more clearly identifies the "ietf-" and "iana-" modules defined.

* Clarified that the modules, when implemented, do not define any protocol-accessible nodes.

* Clarified that IANA may deprecate and/or obsolete identities over time.

* Added Security Consideration for the "generate-asymmetric-key-pair" RPC.

* Added Security Considerations text to also look a SC-section from imported modules.

* Fixed private-key "must" expressions to not require public-key nodes to be present.

* Renamed leaf from "bits" to "num-bits".

* Renamed leaf from "hide" to "hidden".

* Added container "private-key-encoding" to wrap existing choice.

* Removed "public-key-format" and "public-key" nodes from examples.

B.35. 34 to 35

* Addresses AD review by Rob Wilton.

B.36. 35 to 36

* Addresses 1st-round of IESG reviews.

B.37. 36 to 38

* Addresses issues found in OpsDir review of the ssh-client-server draft.

* Replaced identities with enums in the IANA modules.

* Updated per Elwyn Davies' Gen-ART review.

* Updated Introduction to read more like the Abstract

* Add refs to where the 'operational' and 'system' datastores are defined.

* Updated Editor-notes to NOT remove the script (just remove the initial IANA modules)

* Renamed Security Considerations section s/Template for/ Considerations for/

* s/defines/presents/ in a few places.

* Renamed script from 'gen-identities.py' to 'gen-yang-modules.py'

* Removed the removeInRFC="true" attribute in Appendix sections

B.38. 38 to 39

* Address IESG review comments.

B.39. 39 to 40

* Updated to reflect comments from Paul Wouters.

* Fixed the "generate-asymmetric-key-pair" RPC to return the location to where hidden keys are created.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balázs Kovács, Barry Leiba, Benoit Claise, Bert Wijnen, David Lamparter, Elwyn Davies, Gary Wu, Jürgen Schönwälder, Ladislav Lhotka, Liang Xia, Martin Björklund, Martin Thomson, Mehmet Ersue, Michal Vako, Murray Kucherawy, Paul Wouters, Per Andersson, Phil Shafer, Qin Wun, Radek Krejci, Rob Wilton, Roman Danyliw, Russ Housley, Sean Turner, Tom Petch, Thomas Martin, and Warren Kumari.

Contributors

Special acknowledgement goes to Gary Wu for his work on the "ietf-ssh-common" module.

Author's Address

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

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 January 2021

K. Watsen
Watsen Networks
M. Scharf
Hochschule Esslingen

8 July 2020

YANG Groupings for TCP Clients and TCP Servers
draft-ietf-netconf-tcp-client-server-07

Abstract

This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers, either as standalone or in conjunction with a stack protocol layer specific configurations.

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:

* "DDDD" --> the assigned RFC value for this draft

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

* "2020-07-08" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

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

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

This Internet-Draft will expire on 9 January 2021.

Copyright Notice

Table of Contents

1. Introduction

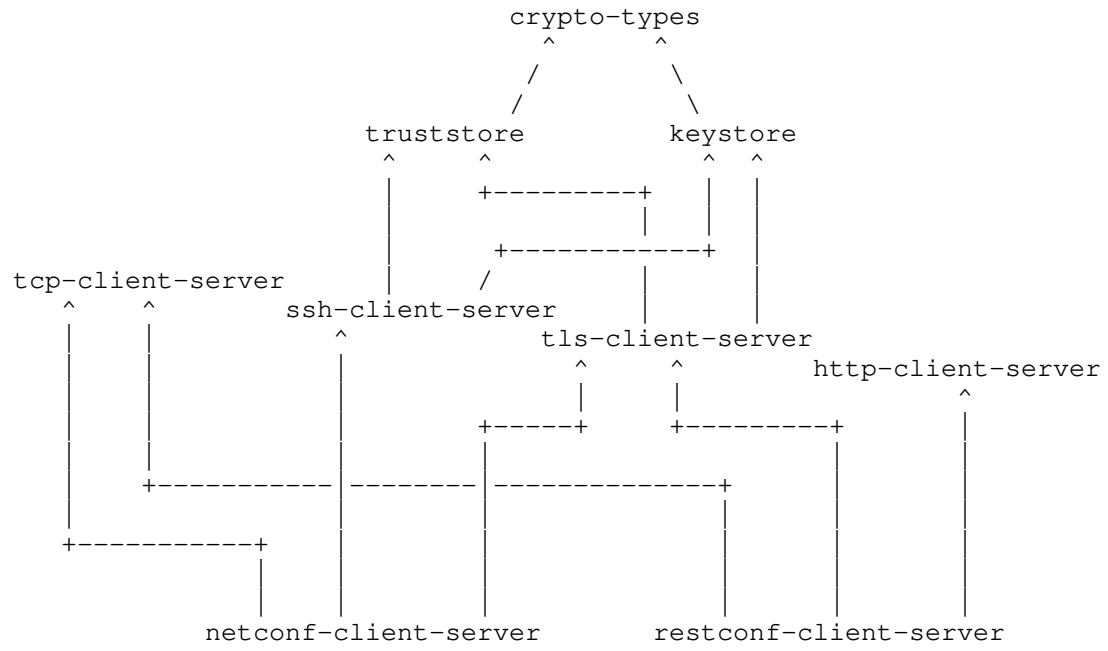
This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers, either as standalone or in conjunction with a stack protocol layer specific configurations.

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

2. The "ietf-tcp-common" Module

2.1. Data Model Overview

2.1.1. Model Scope

This document defines a common "grouping" statement for basic TCP connection parameters that matter to applications. In some TCP stacks, such parameters can also directly be set by an application using system calls, such as the socket API. The base YANG model in this document focuses on modeling TCP keep-alives. This base model can be extended as needed.

2.1.2. Features

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

```
Features:
  +-- keepalives-supported
```

2.1.3. Groupings

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

```
Groupings:
  +-- tcp-common-grouping
  +-- tcp-connection-grouping
```

Each of these groupings are presented in the following subsections.

2.1.3.1. The "tcp-common-grouping" Grouping

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

```
  grouping tcp-common-grouping
    +-- keepalives! {keepalives-supported}?
       +-- idle-time         uint16
       +-- max-probes        uint16
       +-- probe-interval    uint16
```

Comments:

* The "keepalives" node is a "presence" node so that the decendent nodes' "mandatory true" doesn't imply that keepalives must be configured.

* The "idle-time", "max-probes", and "probe-interval" nodes have the common meanings. Please see the YANG module in Section 2.3 for details.

2.1.3.2. The "tcp-connection-grouping" Grouping

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

```
  grouping tcp-connection-grouping
    +---u tcp-common-grouping
```

Comments:

* This grouping uses the "tcp-common-grouping" grouping discussed in Section 2.1.3.1.

2.1.4. Protocol-accessible Nodes

The "ietf-tcp-common" module does not contain any protocol-accessible nodes.

2.1.5. Guidelines for Configuring TCP Keep-Alives

Network stacks may include "keep-alives" in their TCP implementations, although this practice is not universally accepted. If keep-alives are included, [RFC1122] [RFC793bis] mandates that the application MUST be able to turn them on or off for each TCP connection, and that they MUST default to off.

Keep-alive mechanisms exist in many protocols. Depending on the protocol stack, TCP keep-alives may only be one out of several alternatives. Which mechanism(s) to use depends on the use case and application requirements. If keep-alives are needed by an application, it is RECOMMENDED that the aliveness check happens only at the protocol layers that are meaningful to the application.

A TCP keep-alive mechanism SHOULD only be invoked in server applications that might otherwise hang indefinitely and consume resources unnecessarily if a client crashes or aborts a connection during a network failure [RFC1122]. TCP keep-alives may consume significant resources both in the network and in endpoints (e.g., battery power). In addition, frequent keep-alives risk network congestion. The higher the frequency of keep-alives, the higher the overhead.

Given the cost of keep-alives, parameters have to be configured carefully:

* The default idle interval (leaf "idle-time") MUST default to no less than two hours, i.e., 7200 seconds [RFC1122]. A lower value MAY be configured, but keep-alive messages SHOULD NOT be transmitted more frequently than once every 15 seconds. Longer intervals SHOULD be used when possible.

* The maximum number of sequential keep-alive probes that can fail (leaf "max-probes") trades off responsiveness and robustness against packet loss. ACK segments that contain no data are not reliably transmitted by TCP. Consequently, if a keep-alive mechanism is implemented it MUST NOT interpret failure to respond to any specific probe as a dead connection [RFC1122]. Typically a single-digit number should suffice.

* TCP implementations may include a parameter for the number of seconds between TCP keep-alive probes (leaf "probe-interval"). In order to avoid congestion, the time interval between probes MUST NOT be smaller than one second. Significantly longer intervals SHOULD be used. It is important to note that keep-alive probes (or replies) can get dropped due to network congestion. Sending further probe messages into a congested path after a short interval, without backing off timers, could cause harm and result in a congestion collapse. Therefore it is essential to pick a large, conservative value for this interval.

2.2. Example Usage

This section presents an example showing the "tcp-common-grouping" populated with some data.

```
   <tcp-common xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-common">
     <keepalives>
       <idle-time>15</idle-time>
       <max-probes>3</max-probes>
       <probe-interval>30</probe-interval>
     </keepalives>
   </tcp-common>
```

2.3. YANG Module

The ietf-tcp-common YANG module references [RFC6991].

```
<CODE BEGINS> file "ietf-tcp-common@2020-07-08.yang"

module ietf-tcp-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-common";
  prefix tcpcmn;

  organization
    "IETF NETCONF (Network Configuration) Working Group and the
     IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
               <http://datatracker.ietf.org/wg/tcpm/>
     WG List:  <mailto:netconf@ietf.org>
               <mailto:tcpm@ietf.org>
     Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
               Michael Scharf
               <mailto:michael.scharf@hs-esslingen.de>";

  description
    "This module defines reusable groupings for TCP commons that
     can be used as a basis for specific TCP common instances.

     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 DDDD
     (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
```

```
        itself for full legal notices.

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

      revision 2020-07-08 {
        description
          "Initial version";
        reference
          "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
      }

      // Features
      feature keepalives-supported {
        description
          "Indicates that keepalives are supported.";
      }

      // Groupings

      grouping tcp-common-grouping {
        description
          "A reusable grouping for configuring TCP parameters common
           to TCP connections as well as the operating system as a
           whole.";
        container keepalives {
          if-feature "keepalives-supported";
          presence
           "Indicates that keepalives are enabled.  Present so that
            the decendant nodes' 'mandatory true' doesn't imply that
            this node  must be configured.";
          description
            "Configures the keep-alive policy, to proactively test the
             aliveness of the TCP peer.  An unresponsive TCP peer is
             dropped after approximately (idle-time + max-probes
             * probe-interval) seconds.";
          leaf idle-time {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            mandatory true;
            description
              "Sets the amount of time after which if no data has been
```

```
              received from the TCP peer, a TCP-level probe message
              will be sent to test the aliveness of the TCP peer.
              Two hours (7200 seconds) is safe value, per RFC 1122.";
            reference
              "RFC 1122:
               Requirements for Internet Hosts -- Communication Layers";
          }
          leaf max-probes {
            type uint16 {
              range "1..max";
            }
            mandatory true;
            description
              "Sets the maximum number of sequential keep-alive probes
               that can fail to obtain a response from the TCP peer
               before assuming the TCP peer is no longer alive.";
          }
          leaf probe-interval {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            mandatory true;
            description
              "Sets the time interval between failed probes. The interval
               SHOULD be significantly longer than one second in order to
               avoid harm on a congested link.";
          }
        } // container keepalives
      } // grouping tcp-common-grouping


      grouping tcp-connection-grouping {
        description
          "A reusable grouping for configuring TCP parameters common
           to TCP connections.";
         uses tcp-common-grouping;
      }

    }

    <CODE ENDS>
```

3. The "ietf-tcp-client" Module

3.1. Data Model Overview

3.1.1. Features

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

```
Features:
  +-- local-binding-supported
  +-- tcp-client-keepalives
  +-- proxy-connect
  +-- socks5-gss-api
  +-- socks5-username-password
```

3.1.2. Groupings

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

```
Groupings:
  +-- tcp-client-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "tcp-client-grouping" Grouping

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

```
  grouping tcp-client-grouping
    +-- remote-address                    inet:host
    +-- remote-port?                      inet:port-number
    +-- local-address?                    inet:ip-address
    |       {local-binding-supported}?
    +-- local-port?                       inet:port-number
    |       {local-binding-supported}?
    +-- proxy-server! {proxy-connect}?
    |  +-- (proxy-type)
    |     +--:(socks4)
    |     |  +-- socks4-parameters
    |     |     +-- remote-address    inet:ip-address
    |     |     +-- remote-port?      inet:port-number
    |     +--:(socks4a)
    |     |  +-- socks4a-parameters
    |     |     +-- remote-address    inet:host
    |     |     +-- remote-port?      inet:port-number
    |     +--:(socks5)
    |        +-- socks5-parameters
    |           +-- remote-address               inet:host
    |           +-- remote-port?                 inet:port-number
    |           +-- authentication-parameters!
    |              +-- (auth-type)
    |                 +--:(gss-api) {socks5-gss-api}?
    |                 |  +-- gss-api
    |                 +--:(username-password)
    |                           {socks5-username-password}?
    |                    +-- username-password
    |                       +-- username?   string
    |                       +-- password?   string
    |
    +---u tcpcmn:tcp-connection-grouping
```

Comments:

* The "remote-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, a hostname.

* The "remote-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* The "local-address" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), is not mandatory. Its default value is '0', indicating that the operating system can pick an arbitrary port number.

* The "proxy-server" node is enabled by a "feature" statement and, for servers that enable it, is a "presence" container so that the decendent "mandatory true" choice node doesn't imply that the proxt-server node must be configured.

* This grouping uses the "tcp-connection-grouping" grouping discussed in Section 2.1.3.2.

3.1.3. Protocol-accessible Nodes

The "ietf-tcp-client" module does not contain any protocol-accessible nodes.

3.2. Example Usage

This section presents an example showing the "tcp-client-grouping" populated with some data.

```
<tcp-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-client">
  <remote-address>www.example.com</remote-address>
  <remote-port>443</remote-port>
  <local-address>0.0.0.0</local-address>
  <local-port>0</local-port>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-client>
```

3.3. YANG Module

The ietf-tcp-client YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-client@2020-07-08.yang"

```
module ietf-tcp-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-client";
  prefix tcpc;

  import ietf-inet-types {
    prefix inet;
```

```
      reference
        "RFC 6991: Common YANG Data Types";
    }

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

    import ietf-tcp-common {
      prefix tcpcmn;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group and the
       IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

    contact
      "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
                 <http://datatracker.ietf.org/wg/tcpm/>
       WG List:  <mailto:netconf@ietf.org>
                 <mailto:tcpm@ietf.org>
       Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                 Michael Scharf
                 <mailto:michael.scharf@hs-esslingen.de>";

    description
      "This module defines reusable groupings for TCP clients that
       can be used as a basis for specific TCP client instances.

       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 DDDD
       (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
       itself for full legal notices.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
```

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

    revision 2020-07-08 {
      description
        "Initial version";
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    // Features

    feature local-binding-supported {
      description
        "Indicates that the server supports configuring local
         bindings (i.e., the local address and local port) for
         TCP clients.";
    }

    feature tcp-client-keepalives {
      description
        "Per socket TCP keepalive parameters are configurable for
         TCP clients on the server implementing this feature.";
    }

    feature proxy-connect {
      description
        "Proxy connection configuration is configurable for
         TCP clients on the server implementing this feature.";
    }

    feature socks5-gss-api {
      description
        "Indicates that the server supports authenticating
         using GSSAPI when initiating TCP connections via
         and SOCKS Version 5 proxy server.";
      reference
        "RFC 1928: SOCKS Protocol Version 5";
    }

    feature socks5-username-password {
      description
        "Indicates that the server supports authenticating
         using username/password when initiating TCP
         connections via and SOCKS Version 5 proxy
```

```
          server.";
      reference
        "RFC 1928: SOCKS Protocol Version 5";
    }

    // Groupings

    grouping tcp-client-grouping {
      description
        "A reusable grouping for configuring a TCP client.

        Note that this grouping uses fairly typical descendent
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tcp-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      leaf remote-address {
        type inet:host;
        mandatory true;
        description
          "The IP address or hostname of the remote peer to
           establish a connection with.  If a domain name is
           configured, then the DNS resolution should happen on
           each connection attempt.  If the DNS resolution
           results in multiple IP addresses, the IP addresses
           are tried according to local preference order until
           a connection has been established or until all IP
           addresses have failed.";
      }
      leaf remote-port {
        type inet:port-number;
        default "0";
        description
          "The IP port number for the remote peer to establish a
           connection with.  An invalid default value (0) is used
           (instead of 'mandatory true') so that as application
           level data model may 'refine' it with an application
           specific default port number value.";
      }
      leaf local-address {
        if-feature "local-binding-supported";
        type inet:ip-address;
        description
          "The local IP address/interface (VRF?) to bind to for when
```

```
            connecting to the remote peer.  INADDR_ANY ('0.0.0.0') or
            INADDR6_ANY ('0:0:0:0:0:0:0:0' a.k.a. '::') MAY be used to
            explicitly indicate the implicit default, that the server
            can bind to any IPv4 or IPv6 addresses, respectively.";
        }
        leaf local-port {
          if-feature "local-binding-supported";
          type inet:port-number;
          default "0";
          description
            "The local IP port number to bind to for when connecting
             to the remote peer.  The port number '0', which is the
             default value, indicates that any available local port
             number may be used.";
        }

        container proxy-server {
          if-feature "proxy-connect";
          presence
            "Indicates that a proxy connection is configured.
             Present so that the 'proxy-type' node's 'mandatory
             true' doesn't imply that the proxy connection
             must be configured.";
          choice proxy-type {
            mandatory true;
            description
              "Selects a proxy connection protocol.";
            case socks4 {
              container socks4-parameters {
                leaf remote-address {
                  type inet:ip-address;
                  mandatory true;
                  description
                    "The IP address of the proxy server.";
                }
                leaf remote-port {
                  type inet:port-number;
                  default "1080";
                  description
                    "The IP port number for the proxy server.";
                }
                description
                  "Parameters for connecting to a TCP-based proxy
                   server using the SOCKS4 protocol.";
                reference
                  "SOCKS, Proceedings: 1992 Usenix Security Symposium.";
              }
            }
```

```
            case socks4a {
              container socks4a-parameters {
                leaf remote-address {
                  type inet:host;
                  mandatory true;
                  description
                    "The IP address or hostname of the proxy server.";
                }
                leaf remote-port {
                  type inet:port-number;
                  default "1080";
                  description
                    "The IP port number for the proxy server.";
                }
                description
                  "Parameters for connecting to a TCP-based proxy
                   server using the SOCKS4a protocol.";
                reference
                  "SOCKS Proceedings:
                     1992 Usenix Security Symposium.
                   OpenSSH message:
                     SOCKS 4A: A Simple Extension to SOCKS 4 Protocol
                     https://www.openssh.com/txt/socks4a.protocol";
              }
            }
            case socks5 {
              container socks5-parameters {
                leaf remote-address {
                  type inet:host;
                  mandatory true;
                  description
                    "The IP address or hostname of the proxy server.";
                }
                leaf remote-port {
                  type inet:port-number;
                  default "1080";
                  description
                    "The IP port number for the proxy server.";
                }
                container authentication-parameters {
                  presence
                    "Indicates that an authentication mechanism
                     has been configured.  Present so that the
                     'auth-type' node's 'mandatory true' doesn't
                     imply that an authentication mechanism
                     must be configured.";
                  description
                    "A container for SOCKS Version 5 authentication
```

```
          mechanisms.

          A complete list of methods is defined at:
          https://www.iana.org/assignments/socks-methods
          /socks-methods.xhtml.";
        reference
          "RFC 1928: SOCKS Protocol Version 5";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst supported SOCKS Version 5
             authentication mechanisms.";
          case gss-api {
            if-feature socks5-gss-api;
            container gss-api {
              description
                "Contains GSS-API configuration.  Defines
                 as an empty container to enable specific
                 GSS-API configuration to be augmented in
                 by future modules.";
              reference
                "RFC 1928: SOCKS Protocol Version 5
                 RFC 2743: Generic Security Service
                           Application Program Interface
                           Version 2, Update 1";
            }
          }
          case username-password {
            if-feature socks5-username-password;
            container username-password {
              leaf username {
                type string;
                description
                  "The 'username' value to use.";
              }
              leaf password {
                nacm:default-deny-all;
                type string;
                description
                  "The 'password' value to use.";
              }
              description
                "Contains Username/Password configuration.";
              reference
                "RFC 1929: Username/Password Authentication
                           for SOCKS V5";
            }
          }
```

```
                }
              }
              description
                "Parameters for connecting to a TCP-based proxy server
                 using the SOCKS5 protocol.";
              reference
                "RFC 1928: SOCKS Protocol Version 5";
            }
          }
        }
        description
          "Proxy server settings.";
      }

      uses tcpcmn:tcp-connection-grouping {
        augment "keepalives" {
          if-feature "tcp-client-keepalives";
          description
            "Add an if-feature statement so that implementations
             can choose to support TCP client keepalives.";
        }
      }
    }
  }
```

<CODE ENDS>

4. The "ietf-tcp-server" Module

4.1. Data Model Overview

4.1.1. Features

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

```
Features:
  +-- tcp-server-keepalives
```

4.1.2. Groupings

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

```
Groupings:
  +-- tcp-server-grouping
```

Each of these groupings are presented in the following subsections.

4.1.2.1. The "tcp-server-grouping" Grouping

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

```
  grouping tcp-server-grouping
    +-- local-address                    inet:ip-address
    +-- local-port?                      inet:port-number
    +---u tcpcmn:tcp-connection-grouping
```

Comments:

* The "local-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* This grouping uses the "tcp-connection-grouping" grouping discussed in Section 2.1.3.2.

4.1.3. Protocol-accessible Nodes

The "ietf-tcp-server" module does not contain any protocol-accessible nodes.

4.2. Example Usage

This section presents an example showing the "tcp-server-grouping" populated with some data.

```
<tcp-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-server">
  <local-address>10.20.30.40</local-address>
  <local-port>7777</local-port>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-server>
```

4.3. YANG Module

The ietf-tcp-server YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-server@2020-07-08.yang"

```
   module ietf-tcp-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-server";
     prefix tcps;

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

     import ietf-tcp-common {
       prefix tcpcmn;
       reference
         "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group and the
        IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

     contact
       "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
                  <http://datatracker.ietf.org/wg/tcpm/>
        WG List:  <mailto:netconf@ietf.org>
                  <mailto:tcpm@ietf.org>
        Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                  Michael Scharf
                  <mailto:michael.scharf@hs-esslingen.de>";

     description
       "This module defines reusable groupings for TCP servers that
        can be used as a basis for specific TCP server instances.

        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 DDDD
        (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
        itself for full legal notices.
```

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

      revision 2020-07-08 {
        description
          "Initial version";
        reference
          "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
      }

      // Features

      feature tcp-server-keepalives {
        description
          "Per socket TCP keepalive parameters are configurable for
           TCP servers on the server implementing this feature.";
      }


      // Groupings

      grouping tcp-server-grouping {
        description
          "A reusable grouping for configuring a TCP server.

           Note that this grouping uses fairly typical descendent
           node names such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue (e.g., by wrapping
           the 'uses' statement in a container called
           'tcp-server-parameters').  This model purposely does
           not do this itself so as to provide maximum flexibility
           to consuming models.";
        leaf local-address {
          type inet:ip-address;
          mandatory true;
          description
            "The local IP address to listen on for incoming
             TCP client connections.  INADDR_ANY (0.0.0.0) or
             INADDR6_ANY (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be
             used when the server is to listen on all IPv4 or
             IPv6 addresses, respectively.";
        }
        leaf local-port {
```

```
        type inet:port-number;
        default "0";
        description
          "The local port number to listen on for incoming TCP
           client connections.  An invalid default value (0)
           is used (instead of 'mandatory true') so that an
           application level data model may 'refine' it with
           an application specific default port number value.";
      }
      uses tcpcmn:tcp-connection-grouping {
        augment "keepalives" {
          if-feature "tcp-server-keepalives";
          description
            "Add an if-feature statement so that implementations
             can choose to support TCP server keepalives.";
        }
      }
    }
  }
```

<CODE ENDS>

5. Security Considerations

5.1. The "ietf-tcp-common" YANG Module

The "ietf-tcp-common" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. The "ietf-tcp-client" YANG Module

The "ietf-tcp-client" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "proxy-server/socks5-parameters/authentication-parameters/username-password/password" node:

  The cleartext "password" node defined in the "tcp-client-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.3. The "ietf-tcp-server" YANG Module

The "ietf-tcp-server" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-tcp-client
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tcp-server
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

6.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-tcp-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-common
prefix:       tcpcmn
reference:    RFC DDDD

name:         ietf-tcp-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-client
prefix:       tcpc
reference:    RFC DDDD

name:         ietf-tcp-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-server
prefix:       tcps
reference:    RFC DDDD
```

7. References

7.1. 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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[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>.

7.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

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

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

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

[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>.

Appendix A. Change Log

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

A.1. 00 to 01

* Added 'local-binding-supported' feature to TCP-client model.

* Added 'keepalives-supported' feature to TCP-common model.

* Added 'external-endpoint-values' container and 'external-endpoints' feature to TCP-server model.

A.2. 01 to 02

* Removed the 'external-endpoint-values' container and 'external-endpoints' feature from the TCP-server model.

A.3. 02 to 03

* Moved the common model section to be before the client and server specific sections.

* Added sections "Model Scope" and "Usage Guidelines for Configuring TCP Keep-Alives" to the common model section.

A.4. 03 to 04

* Fixed a few typos.

A.5. 04 to 05

* Removed commented out "grouping tcp-system-grouping" statement kept for reviewers.

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

A.6. 05 to 06

* Added support for TCP proxies.

A.7. 06 to 07

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

* Updated the Security Considerations section.

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Michael Scharf
Hochschule Esslingen - University of Applied Sciences

Email: michael.scharf@hs-esslingen.de

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 October 2024

K. Watsen
Watsen Networks
M. Scharf
Hochschule Esslingen

4 April 2024

YANG Groupings for TCP Clients and TCP Servers
draft-ietf-netconf-tcp-client-server-26

Abstract

This document presents three YANG 1.1 modules to support the configuration of TCP clients and TCP servers. The modules include basic parameters of a TCP connection relevant for client or server applications, as well as client configuration required for traversing proxies. The modules can be used either standalone or in conjunction with configuration of other stack protocol layers.

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

* DDDD --> 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:

* 2024-04-04 --> the publication date of this draft

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

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers (TCP is defined in [RFC9293]), either as standalone or in conjunction with configuration of other stack protocol layers.

The modules focus on three different types of base TCP parameters that matter for TCP-based applications: First, the modules cover fundamental configuration of a TCP client or TCP server application, such as addresses and port numbers. Second, a reusable grouping enables modification of application-specific parameters for a TCP connections, such as use of TCP keep-alives. And third, client configuration for traversing proxies is included as well. In each case, the modules have a very narrow scope and focus on a minimum set of required parameters.

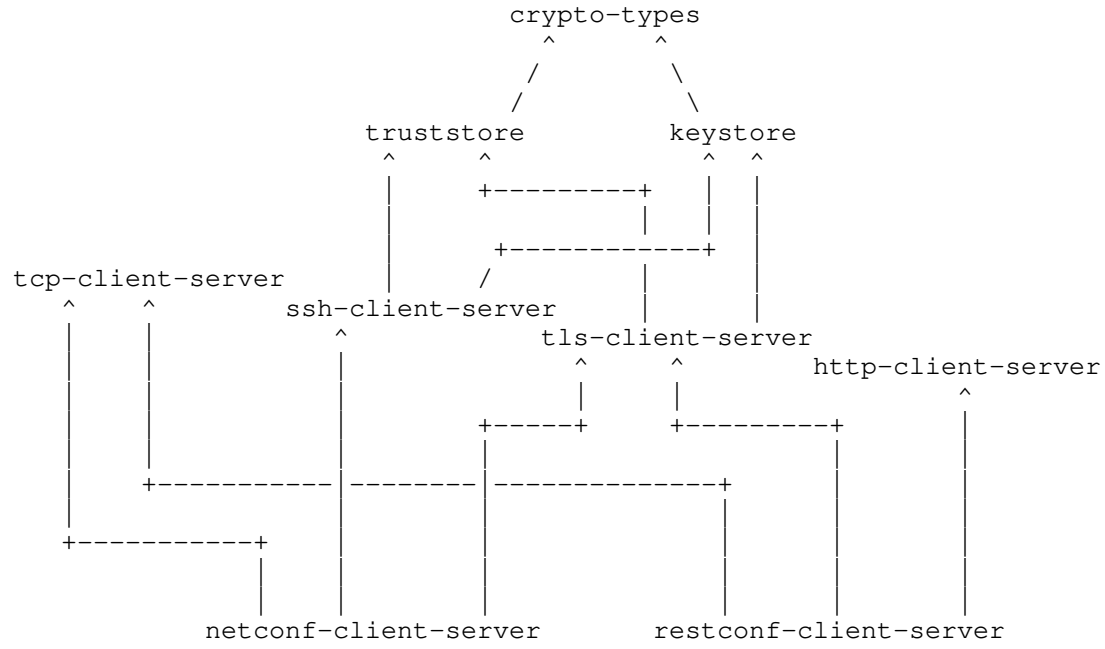
Please be advised that while this document presents support for some TCP proxy techniques, there are other TCP proxy techniques that are not part of this document, but could be added by augmenting the YANG module.

1.1. Relation to other RFCs

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

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



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

Table 1: Label 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]. It does not define any protocol accessible nodes that are "config false".

2. The "ietf-tcp-common" Module

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

2.1. Data Model Overview

This section provides an overview of the "ietf-tcp-common" module in terms of its features and groupings.

2.1.1. Model Scope

This document presents a common "grouping" statement for basic TCP connection parameters that matter to applications. It is "common" in that this grouping is used by both the "ietf-tcp-client" and "ietf-tcp-server" modules. In some TCP stacks, such parameters can also directly be set by an application using system calls, such as the sockets API. The base YANG model in this document focuses on modeling TCP keep-alives. This base model can be extended as needed.

2.1.2. Features

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

```
Features:
  +-- keepalives-supported
```

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

2.1.3. Groupings

The "ietf-tcp-common" module defines the following "grouping" statement:

* tcp-common-grouping

This grouping is presented in the following subsection.

2.1.3.1. The "tcp-common-grouping" Grouping

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

```
  grouping tcp-common-grouping:
    +-- keepalives! {keepalives-supported}?
       +-- idle-time?        uint16
       +-- max-probes?       uint16
       +-- probe-interval?   uint16
```

Comments:

* The "keepalives" node is a "presence" container so that the mandatory descendant nodes do not imply that keepalives must be configured.

* The "idle-time", "max-probes", and "probe-interval" nodes have the common meanings. Please see the YANG module in Section 2.3 for details.

2.1.4. Protocol-accessible Nodes

The "ietf-tcp-common" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

2.1.5. Guidelines for Configuring TCP Keep-Alives

Network stacks may include "keep-alives" in their TCP implementations, although this practice is not universally implemented. If keep-alives are included, [RFC9293] mandates that the application MUST be able to turn them on or off for each TCP connection, and that they MUST default to off.

Keep-alive mechanisms exist in many protocols. Depending on the protocol stack, TCP keep-alives may only be one out of several alternatives. Which mechanism(s) to use depends on the use case and application requirements. If keep-alives are needed by an application, it is RECOMMENDED that the liveness check happens only at the protocol layers that are meaningful to the application.

A TCP keep-alive mechanism SHOULD only be invoked in server applications that might otherwise hang indefinitely and consume resources unnecessarily if a client crashes or aborts a connection during a network failure [RFC9293]. TCP keep-alives may consume significant resources both in the network and in endpoints (e.g., battery power). In addition, frequent keep-alives risk network congestion. The higher the frequency of keep-alives, the higher the overhead.

Given the cost of keep-alives, parameters have to be configured carefully:

* The default idle interval (leaf "idle-time") is two hours, i.e., 7200 seconds [RFC9293]. A lower value MAY be configured, but idle intervals SHOULD NOT be smaller than 15 seconds. Longer idle intervals SHOULD be used when possible.

* The maximum number of sequential keep-alive probes that can fail (leaf "max-probes") trades off responsiveness and robustness against packet loss. ACK segments that contain no data are not reliably transmitted by TCP. Consequently, if a keep-alive mechanism is implemented it MUST NOT interpret failure to respond to any specific probe as a dead connection [RFC9293]. Typically, a single-digit number should suffice.

* TCP implementations may include a parameter for the number of seconds between TCP keep-alive probes (leaf "probe-interval"). In order to avoid congestion, the time interval between probes MUST NOT be smaller than one second. Significantly longer intervals SHOULD be used. It is important to note that keep-alive probes (or replies) can get dropped due to network congestion. Sending further probe messages into a congested path after a short interval, without backing off timers, could cause harm and result in a congestion collapse. Therefore it is essential to pick a large, conservative value for this interval.

2.2. Example Usage

This section presents an example showing the "tcp-common-grouping" populated with some data.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-common xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-common">
  <keepalives>
    <idle-time>7200</idle-time>
    <max-probes>9</max-probes>
    <probe-interval>75</probe-interval>
  </keepalives>
</tcp-common>
```

2.3. YANG Module

The ietf-tcp-common YANG module references [RFC6991] and [RFC9293].

<CODE BEGINS> file "ietf-tcp-common@2024-04-04.yang"

```
module ietf-tcp-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-common";
  prefix tcpcmn;

  organization
    "IETF NETCONF (Network Configuration) Working Group and the
```

```
      IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

    contact
      "WG Web:   https://datatracker.ietf.org/wg/netconf
                 https://datatracker.ietf.org/wg/tcpm
       WG List:  NETCONF WG list <mailto:netconf@ietf.org>
                 TCPM WG list <mailto:tcpm@ietf.org>
       Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                 Michael Scharf
                 <mailto:michael.scharf@hs-esslingen.de>";

    description
      "This module define a reusable 'grouping' that is common
       to both TCP-clients and TCP-servers.  This grouping statement
       is used by both the 'ietf-tcp-client' and 'ietf-tcp-server'
       modules.

       Copyright (c) 2023 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 DDDD
       (https://www.rfc-editor.org/info/rfcDDDD); 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 2024-04-04 {
      description
        "Initial version";
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    // Features

    feature keepalives-supported {
```

```
      description
        "Indicates that keepalives are supported.";
    }

    // Groupings

    grouping tcp-common-grouping {
      description
        "A reusable grouping for configuring TCP parameters common
         to TCP connections as well as the operating system as a
         whole.";
      container keepalives {
        if-feature "keepalives-supported";
        presence
          "Indicates that keepalives are enabled, aligning to
           the requirement in Section 3.8.4 RFC 9293 that
           keepalives are off by default.";
        description
          "Configures the keep-alive policy, to proactively test the
           aliveness of the TCP peer.  An unresponsive TCP peer is
           dropped after approximately (idle-time + max-probes *
           probe-interval) seconds.  Further guidance can be found
           in Section 2.1.5 of RFC DDDD.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP)";
        leaf idle-time {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default 7200;
          description
            "Sets the amount of time after which if no data has been
             received from the TCP peer, a TCP-level probe message
             will be sent to test the aliveness of the TCP peer.
             Two hours (7200 seconds) is safe value, per RFC 9293
             Section 3.8.4.";
          reference
            "RFC 9293: Transmission Control Protocol (TCP)";
        }
        leaf max-probes {
          type uint16 {
            range "1..max";
          }
          default 9;
          description
            "Sets the maximum number of sequential keep-alive probes
             that can fail to obtain a response from the TCP peer
```

```
            before assuming the TCP peer is no longer alive.";
        }
        leaf probe-interval {
          type uint16 {
            range "1..max";
          }
          units "seconds";
          default 75;
          description
            "Sets the time interval between failed probes. The interval
             SHOULD be significantly longer than one second in order to
             avoid harm on a congested link.";
        }
      } // container keepalives
    } // grouping tcp-common-grouping

  }
```

<CODE ENDS>

3. The "ietf-tcp-client" Module

This section defines a YANG 1.1 module called "ietf-tcp-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-tcp-client" module in terms of its features and groupings.

3.1.1. Features

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

```
Features:
  +-- local-binding-supported
  +-- tcp-client-keepalives
  +-- proxy-connect
      +-- socks4-supported {proxy-connect}?
      +-- socks4a-supported {proxy-connect}?
      +-- socks5-supported {proxy-connect}?
          +-- socks5-gss-api {socks5-supported}?
          +-- socks5-username-password {socks5-supported}?
```

Comments:

* The "local-binding-supported" feature indicates that the server supports configuring local bindings (i.e., the local address and local port) for TCP clients."

* The "tcp-client-keepalives" feature indicates that per socket TCP keepalive parameters are configurable for TCP clients on the server implementing this feature.

* The "proxy-connect" feature indicates the TCP-client supports connecting through TCP proxies.

* The "socks4-supported" feature indicates the TCP-client supports Socks4-based proxies.

* The "socks4a-supported" feature indicates the TCP-client supports Socks4a-based proxies. The difference between Socks4 and Socks4a is that Socks4a enables the "remote-address" to be specified using a hostname, in addition to an IP address.

* The "socks5-supported" feature indicates the TCP-client supports Socks5-based proxies.

* The "socks5-gss-api" feature indicates that the server, when acting as a TCP-client, supports authenticating to a SOCKS Version 5 proxy server using GSSAPI credentials.

* The "socks5-username-password" feature indicates that the server, when acting as a TCP-client, supports authenticating to a SOCKS Version 5 proxy server using 'username' and 'password' credentials."

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

3.1.2. Groupings

The "ietf-tcp-client" module defines the following "grouping" statement:

* tcp-client-grouping

This grouping is presented in the following subsection.

3.1.2.1. The "tcp-client-grouping" Grouping

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

```
grouping tcp-client-grouping:
  +-- remote-address                inet:host
  +-- remote-port?                  inet:port-number
  +-- local-address?                inet:ip-address
  |       {local-binding-supported}?
  +-- local-port?                   inet:port-number
  |       {local-binding-supported}?
  +-- proxy-server! {proxy-connect}?
  |  +-- (proxy-type)
  |     +--:(socks4) {socks4-supported}?
  |     |  +-- socks4-parameters
  |     |     +-- remote-address    inet:ip-address
  |     |     +-- remote-port?      inet:port-number
  |     +--:(socks4a) {socks4a-supported}?
  |     |  +-- socks4a-parameters
  |     |     +-- remote-address    inet:host
  |     |     +-- remote-port?      inet:port-number
  |     +--:(socks5) {socks5-supported}?
  |        +-- socks5-parameters
  |           +-- remote-address               inet:host
  |           +-- remote-port?                 inet:port-number
  |           +-- authentication-parameters!
  |              +-- (auth-type)
  |                 +--:(gss-api) {socks5-gss-api}?
  |                 |  +-- gss-api
  |                 +--:(username-password)
  |                           {socks5-username-password}?
  |                    +-- username-password
  |                       +-- username                string
  |                       +---u ct:password-grouping
  |
  +---u tcpcmn:tcp-common-grouping
```

Comments:

* The "remote-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, or a hostname.

* The "remote-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* The "local-address" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), is not mandatory. Its default value is '0', indicating that the operating system can pick an arbitrary port number.

* The "proxy-server" node is enabled by a "feature" statement and, for servers that enable it, is a "presence" container so that the descendant "mandatory true" choice node does not imply that the proxy-server node must be configured. The proxy-server node uses a "choice" statement to allow one of several types of proxies to be configured. The choices presented in this document include Socks4, Socks4a, and Socks5, each enabled by a YANG feature (see Section 3.1.1). Other proxy types may be added by future work.

* This grouping uses the "password-grouping" grouping discussed in [I-D.ietf-netconf-crypto-types].

* This grouping uses the "tcp-common-grouping" grouping discussed in Section 2.1.3.1.

3.1.3. Protocol-accessible Nodes

The "ietf-tcp-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "tcp-client-grouping" populated with some data. This example shows a TCP-client configured to not connect via a proxy:

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-client">
  <remote-address>www.example.com</remote-address>
  <remote-port>8443</remote-port>
  <local-address>192.0.2.2</local-address>
  <local-port>12345</local-port>
  <keepalives>
    <idle-time>7200</idle-time>
    <max-probes>9</max-probes>
    <probe-interval>75</probe-interval>
  </keepalives>
</tcp-client>
```

This example shows a TCP-client configured to connect via a proxy:

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-client">
  <remote-address>www.example.com</remote-address>
  <remote-port>8443</remote-port>
  <local-address>192.0.2.2</local-address>
  <local-port>12345</local-port>
  <proxy-server>
    <socks5-parameters>
      <remote-address>proxy.example.com</remote-address>
      <remote-port>1080</remote-port>
      <authentication-parameters>
        <username-password>
          <username>foobar</username>
          <cleartext-password>secret</cleartext-password>
        </username-password>
      </authentication-parameters>
    </socks5-parameters>
  </proxy-server>
  <keepalives>
    <idle-time>7200</idle-time>
    <max-probes>9</max-probes>
    <probe-interval>75</probe-interval>
  </keepalives>
</tcp-client>
```

3.3. YANG Module

The ietf-tcp-client YANG module references [SOCKS_4A], [RFC1928], [RFC1929], [RFC2743], [RFC6991], [RFC9293], and [I-D.ietf-netconf-crypto-types].

```
<CODE BEGINS> file "ietf-tcp-client@2024-04-04.yang"

module ietf-tcp-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-client";
  prefix tcpc;

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

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

  import ietf-tcp-common {
    prefix tcpcmn;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group and the
     IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
               https://datatracker.ietf.org/wg/tcpm
     WG List:  NETCONF WG list <mailto:netconf@ietf.org>
               TCPM WG list <mailto:tcpm@ietf.org>
     Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
               Michael Scharf
               <mailto:michael.scharf@hs-esslingen.de>";

  description
    "This module defines reusable groupings for TCP clients that
     can be used as a basis for specific TCP client instances.

     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 DDDD
     (https://www.rfc-editor.org/info/rfcDDDD); 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 2024-04-04 {
      description
        "Initial version";
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    // Features

    feature local-binding-supported {
      description
        "Indicates that the server supports configuring local
         bindings (i.e., the local address and local port) for
         TCP clients.";
    }

    feature tcp-client-keepalives {
      description
        "Per socket TCP keepalive parameters are configurable for
         TCP clients on the server implementing this feature.";
      reference
        "RFC 9293: Transmission Control Protocol (TCP)";
    }

    feature proxy-connect {
      description
        "Indicates the TCP-client supports connecting through
         TCP proxies.";
    }

    feature socks4-supported {
      if-feature proxy-connect;
      description
        "Indicates the TCP-client supports Socks4-based proxies.";
      reference
        "SOCKS Proceedings:
           1992 Usenix Security Symposium.";
    }

    feature socks4a-supported {
      if-feature proxy-connect;
      description
        "Indicates the TCP-client supports Socks4a-based proxies.";
      reference
        "OpenSSH message:
           SOCKS 4A: A Simple Extension to SOCKS 4 Protocol
```

```
          https://www.openssh.com/txt/socks4a.protocol.";
      }

      feature socks5-supported {
        if-feature proxy-connect;
        description
          "Indicates the TCP-client supports Socks5-based proxies.";
        reference
          "RFC 1928:
             SOCKS Protocol Version 5";
      }

      feature socks5-gss-api {
        if-feature socks5-supported;
        description
          "Indicates that the server, when acting as a TCP-client,
           supports authenticating to a SOCKS Version 5 proxy server
           using GSSAPI credentials.";
        reference
          "RFC 1928: SOCKS Protocol Version 5";
      }

      feature socks5-username-password {
        if-feature socks5-supported;
        description
          "Indicates that the server, when acting as a TCP-client,
           supports authenticating to a SOCKS Version 5 proxy server
           using 'username' and 'password' credentials.";
        reference
          "RFC 1928: SOCKS Protocol Version 5";
      }

      // Groupings

      grouping tcp-client-grouping {
        description
          "A reusable grouping for configuring a TCP client.

           Note that this grouping uses fairly typical descendant
           node names such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue (e.g., by wrapping
           the 'uses' statement in a container called
           'tcp-client-parameters').  This model purposely does
           not do this itself so as to provide maximum flexibility
           to consuming models.";

        leaf remote-address {
```

```
          type inet:host;
          mandatory true;
          description
            "The IP address or hostname of the remote peer to
             establish a connection with.  If a domain name is
             configured, then the DNS resolution should happen on
             each connection attempt.  If the DNS resolution
             results in multiple IP addresses, the IP addresses
             are tried according to local preference order until
             a connection has been established or until all IP
             addresses have failed.";
        }
        leaf remote-port {
          type inet:port-number;
          default "0";
          description
            "The IP port number for the remote peer to establish a
             connection with.  An invalid default value is used
             so that importing modules may 'refine' it with the
             appropriate default port number value.";
        }
        leaf local-address {
          if-feature "local-binding-supported";
          type inet:ip-address;
          description
            "The local IP address/interface to bind to for when
             connecting to the remote peer.  INADDR_ANY ('0.0.0.0') or
             INADDR6_ANY ('0:0:0:0:0:0:0:0' a.k.a. '::') MAY be used to
             explicitly indicate the implicit default, that the server
             can bind to any IPv4 or IPv6 address.";
        }
        leaf local-port {
          if-feature "local-binding-supported";
          type inet:port-number;
          default "0";
          description
            "The local IP port number to bind to for when connecting
             to the remote peer.  The port number '0', which is the
             default value, indicates that any available local port
             number may be used.";
        }
        container proxy-server {
          if-feature "proxy-connect";
          presence
            "Indicates that a proxy connection has been configured.
             Present so that the mandatory descendant nodes do not
             imply that this node must be configured.";
          choice proxy-type {
```

```
          mandatory true;
          description
            "Selects a proxy connection protocol.";
          case socks4 {
            if-feature socks4-supported;
            container socks4-parameters {
              leaf remote-address {
                type inet:ip-address;
                mandatory true;
                description
                  "The IP address of the proxy server.";
              }
              leaf remote-port {
                type inet:port-number;
                default "1080";
                description
                  "The IP port number for the proxy server.";
              }
              description
                "Parameters for connecting to a TCP-based proxy
                 server using the SOCKS4 protocol.";
              reference
                "SOCKS, Proceedings: 1992 Usenix Security Symposium.";
            }
          }
          case socks4a {
            if-feature socks4a-supported;
            container socks4a-parameters {
              leaf remote-address {
                type inet:host;
                mandatory true;
                description
                  "The IP address or hostname of the proxy server.";
              }
              leaf remote-port {
                type inet:port-number;
                default "1080";
                description
                  "The IP port number for the proxy server.";
              }
              description
                "Parameters for connecting to a TCP-based proxy
                 server using the SOCKS4a protocol.";
              reference
                "SOCKS Proceedings:
                   1992 Usenix Security Symposium.
                 OpenSSH message:
                   SOCKS 4A: A Simple Extension to SOCKS 4 Protocol
```

```
                https://www.openssh.com/txt/socks4a.protocol";
            }
          }
          case socks5 {
            if-feature socks5-supported;
            container socks5-parameters {
              leaf remote-address {
                type inet:host;
                mandatory true;
                description
                  "The IP address or hostname of the proxy server.";
              }
              leaf remote-port {
                type inet:port-number;
                default "1080";
                description
                  "The IP port number for the proxy server.";
              }
              container authentication-parameters {
                presence
                  "Indicates that an authentication mechanism
                   has been configured.  Present so that the
                   mandatory descendant nodes do not imply that
                   this node must be configured.";
                description
                  "A container for SOCKS Version 5 authentication
                   mechanisms.

                   A complete list of methods is defined at:
                   https://www.iana.org/assignments/socks-methods
                   /socks-methods.xhtml.";
                reference
                  "RFC 1928: SOCKS Protocol Version 5";
                choice auth-type {
                  mandatory true;
                  description
                    "A choice amongst supported SOCKS Version 5
                     authentication mechanisms.";
                  case gss-api {
                    if-feature "socks5-gss-api";
                    container gss-api {
                      description
                        "Contains GSS-API configuration.  Defines
                         as an empty container to enable specific
                         GSS-API configuration to be augmented in
                         by future modules.";
                      reference
                        "RFC 1928: SOCKS Protocol Version 5
```

```
                        RFC 2743: Generic Security Service
                                  Application Program Interface
                                  Version 2, Update 1";
                    }
                  }
                  case username-password {
                    if-feature "socks5-username-password";
                    container username-password {
                      leaf username {
                        type string;
                        mandatory true;
                        description
                          "The 'username' value to use for client
                           identification.";
                      }
                      uses ct:password-grouping {
                        description
                          "The password to be used for client
                           authentication.";
                      }
                      description
                        "Contains Username/Password configuration.";
                      reference
                        "RFC 1929: Username/Password Authentication
                                   for SOCKS V5";
                    }
                  }
                }
              }
              description
                "Parameters for connecting to a TCP-based proxy server
                 using the SOCKS5 protocol.";
              reference
                "RFC 1928: SOCKS Protocol Version 5";
            }
          }
        }
        description
          "Proxy server settings.";
      }

      uses tcpcmn:tcp-common-grouping {
        refine "keepalives" {
          if-feature "tcp-client-keepalives";
          description
            "An if-feature statement so that implementations
             can choose to support TCP client keepalives.";
        }
```

```
      }
    }
  }
```

<CODE ENDS>

4. The "ietf-tcp-server" Module

This section defines a YANG 1.1 module called "ietf-tcp-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-tcp-server" module in terms of its features and groupings.

4.1.1. Features

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

```
Features:
  +-- tcp-server-keepalives
```

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

4.1.2. Groupings

The "ietf-tcp-server" module defines the following "grouping" statement:

* tcp-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "tcp-server-grouping" Grouping

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

```
  grouping tcp-server-grouping:
    +-- local-bind* [local-address]
    |  +-- local-address?   inet:ip-address
    |  +-- local-port?      inet:port-number
    +---u tcpcmn:tcp-common-grouping
```

Comments:

* The "local-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* This grouping uses the "tcp-common-grouping" grouping discussed in Section 2.1.3.1.

4.1.3. Protocol-accessible Nodes

The "ietf-tcp-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

4.2. Example Usage

This section presents an example showing the "tcp-server-grouping" populated with some data.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tcp-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-server">
  <local-bind>
    <local-address>192.0.2.2</local-address>
    <local-port>49152</local-port>
  </local-bind>
  <keepalives>
    <idle-time>7200</idle-time>
    <max-probes>9</max-probes>
    <probe-interval>75</probe-interval>
  </keepalives>
</tcp-server>
```

4.3. YANG Module

The ietf-tcp-server YANG module references [RFC6991] and [RFC9293].

<CODE BEGINS> file "ietf-tcp-server@2024-04-04.yang"

```
   module ietf-tcp-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-server";
     prefix tcps;

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

     import ietf-tcp-common {
       prefix tcpcmn;
       reference
         "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group and the
        IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

     contact
       "WG Web:   https://datatracker.ietf.org/wg/netconf
                  https://datatracker.ietf.org/wg/tcpm
        WG List:  NETCONF WG list <mailto:netconf@ietf.org>
                  TCPM WG list <mailto:tcpm@ietf.org>
        Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                  Michael Scharf
                  <mailto:michael.scharf@hs-esslingen.de>";

     description
       "This module defines reusable groupings for TCP servers that
        can be used as a basis for specific TCP server instances.

        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 DDDD
        (https://www.rfc-editor.org/info/rfcDDDD); 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 2024-04-04 {
        description
          "Initial version";
        reference
          "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
      }

      // Features

      feature tcp-server-keepalives {
        description
          "Per socket TCP keepalive parameters are configurable for
           TCP servers on the server implementing this feature.";
        reference
          "RFC 9293: Transmission Control Protocol (TCP)";
      }

      // Groupings

      grouping tcp-server-grouping {
        description
          "A reusable grouping for configuring a TCP server.

           Note that this grouping uses fairly typical descendant
           node names such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue (e.g., by wrapping
           the 'uses' statement in a container called
           'tcp-server-parameters').  This model purposely does
           not do this itself so as to provide maximum flexibility
           to consuming models.";
        list local-bind {
          key local-address;
          min-elements 1;
          description
            "A list of bind (listen) points for this server
             instance.  A server instance may have multiple
             bind points to support, e.g., the same port in
             different address families or different ports
             in the same address family.";
          leaf local-address {
```

```
          type inet:ip-address;
          description
            "The local IP address to listen on for incoming
             TCP client connections.  To configure listening
             on all IPv4 addresses the value must be '0.0.0.0'
             (INADDR_ANY).  To configure listening on all IPv6
             addresses the value must be '::' (INADDR6_ANY).";
        }
        leaf local-port {
          type inet:port-number;
          default "0";
          description
            "The local port number to listen on for incoming TCP
             client connections.  An invalid default value (0)
             is used (instead of 'mandatory true') so that an
             application level data model may 'refine' it with
             an application specific default port number value.";
        }
      }
      uses tcpcmn:tcp-common-grouping {
        refine "keepalives" {
          if-feature "tcp-server-keepalives";
          description
            "An if-feature statement so that implementations
             can choose to support TCP server keepalives.";
        }
      }
    }
  }
```

<CODE ENDS>

5. Security Considerations

The three YANG modules in this document define groupings and will not be deployed as standalone modules. Their security implications may be context dependent based on their use in other modules. The designers of modules which import these grouping must conduct their own analysis of the security considerations.

5.1. Considerations for the "ietf-tcp-common" YANG Module

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

The "ietf-tcp-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. Considerations for the "ietf-tcp-client" YANG Module

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

The "ietf-tcp-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "proxy-server/socks5-parameters/authentication-parameters/username-password/password" node:

  The "password" node defined in the "tcp-client-grouping" grouping is defined using the "password-grouping" grouping presented in [I-D.ietf-netconf-crypto-types]. This grouping enables both cleartext and encrypted passwords to be configured. As the referenced document states, configuration of cleartext passwords is NOT RECOMMENDED. However, in the case cleartext values are configured, this node is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Implementations are RECOMMENDED to implement the "local-binding-supported" feature for cryptographically-secure protocols, so as to enable more granular ingress/egress firewall rulebases. It is NOT RECOMMENDED to implement this feature for unsecure protocols, as per [RFC6056].

5.3. Considerations for the "ietf-tcp-server" YANG Module

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

The "ietf-tcp-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers three URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

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

6.2. The "YANG Module Names" Registry

This document registers three YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-tcp-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-common
prefix:       tcpcmn
reference:    RFC DDDD

name:         ietf-tcp-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-client
prefix:       tcpc
reference:    RFC DDDD

name:         ietf-tcp-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-server
prefix:       tcps
reference:    RFC DDDD
```

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-34, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-34>.

[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>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[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>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-20, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-20>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-35, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-35>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-36, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-36>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-36, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-36>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-40, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-40>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-24, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-24>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-41, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-41>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-28, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-28>.

[RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <https://www.rfc-editor.org/info/rfc1928>.

[RFC1929] Leech, M., "Username/Password Authentication for SOCKS V5", RFC 1929, DOI 10.17487/RFC1929, March 1996, <https://www.rfc-editor.org/info/rfc1929>.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, DOI 10.17487/RFC2743, January 2000, <https://www.rfc-editor.org/info/rfc2743>.

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

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, DOI 10.17487/RFC6056, January 2011, <https://www.rfc-editor.org/info/rfc6056>.

[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>.

[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>.

[SOCKS_4A] Project, T. O., "SOCKS 4A: A Simple Extension to SOCKS 4 Protocol", <https://www.openssh.com/txt/socks4a.protocol>.

Appendix A. Change Log

A.1. 00 to 01

* Added 'local-binding-supported' feature to TCP-client model.

* Added 'keepalives-supported' feature to TCP-common model.

* Added 'external-endpoint-values' container and 'external-endpoints' feature to TCP-server model.

A.2. 01 to 02

* Removed the 'external-endpoint-values' container and 'external-endpoints' feature from the TCP-server model.

A.3. 02 to 03

* Moved the common model section to be before the client and server specific sections.

* Added sections "Model Scope" and "Usage Guidelines for Configuring TCP Keep-Alives" to the common model section.

A.4. 03 to 04

* Fixed a few typos.

A.5. 04 to 05

* Removed commented out "grouping tcp-system-grouping" statement kept for reviewers.

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

A.6. 05 to 06

* Added support for TCP proxies.

A.7. 06 to 07

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

* Updated the Security Considerations section.

A.8. 07 to 08

* Added missing IANA registration for "ietf-tcp-common"

* Added "mandatory true" for the "username" and "password" leafs

* Added an example of a TCP-client configured to connect via a proxy

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

* Updated the "ietf-tcp-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

A.9. 08 to 09

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.10. 09 to 10

* Updated Abstract and Intro to address comments by Tom Petch.
* Removed the "tcp-connection-grouping" grouping (now models use the "tcp-common-grouping" directly).
* Added XML-comment above examples explaining the reason for the unusual top-most element's presence.
* Added Securty Considerations section for the "local-binding-supported" feature.
* Replaced some hardcoded refs to <xref> elements.
* Fixed nits found by YANG Doctor reviews.
* Aligned modules with `pyang -f` formatting.
* Added an "Acknowledgements" secetion.

A.11. 10 to 11

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
* Minor editorial nits

A.12. 11 to 12

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)
* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

A.13. 12 to 13

* NO UPDATE.

A.14. 13 to 14

* Updated per Shepherd reviews impacting the suite of drafts.

A.15. 14 to 15

* Updated per Shepherd reviews impacting the suite of drafts.

A.16. 15 to 16

* Updated per Tom Petch review.
* Added refs to RFC9293 and SOCKS 4A.
* Fixed examples to use IETF-sanctioned values for examples.

A.17. 16 to 17

* Addresses AD review comments.
* Added note to Editor to fix line foldings.
* Added Security Considerations text to also look a SC-section from imported modules.
* Fixed bug: s/augment "keepalives"/refine "keepalives"/
* Set defaults for idle-time, max-probes, and probe-interval (removed "mandatory true").
* Updated examples to use IETF recommended values for examples.

A.18. 18 to 19

* Addresses AD review by Rob Wilton.

A.19. 18 to 19

* Replace RFC 1122 with RFC 9293.

A.20. 19 to 20

* Addresses 1st-round of IESG reviews.

A.21. 20 to 22

* Addresses issues found in OpsDir review of the ssh-client-server draft.
* Updated to address Mallory Knodel's Gen-ART review.
* Renamed Security Considerations section s/Template for/ Considerations for/
* s/defines/presents/ in a few places.

* Add refs to where the 'operational' and 'system' datastores are defined.

* Added more 'feature' statements and descriptions for them

* Added Security Considaration for the "password" node

A.22. 22 to 23

* Address IESG review comments.

A.23. 23 to 24

* Nothing changed. Bumped only for automation.

A.24. 24 to 25

* Updated to support dual-stacks

A.25. 25 to 26

* Updated to ensure at least one local-bind is specified.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Éric Vyncke, Joe Clarke, Jürgen Schönwälder, Ladislav Lhotka, Mallory Knodel, Martin Duke, Michael Tüxen, Mohamed Boucadair, Nancy Cam-Winget, Nick Hancock, Per Andersson, Rob Wilton, Roman Danyliw, Tom Petch, Wim Henderickx.

Authors' Addresses

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

Michael Scharf
Hochschule Esslingen - University of Applied Sciences
Email: michael.scharf@hs-esslingen.de

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

K. Watsen
Watsen Networks
G. Wu
Cisco Systems

10 July 2020

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-21

Abstract

This document defines three YANG modules: the first defines groupings for a generic TLS client, the second defines groupings for a generic TLS server, and the third defines common identities and groupings used by both the client and the server. It is intended that these groupings will be used by applications using the TLS protocol.

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 draft-ietf-netconf-trust-anchors

* "CCCC" --> the assigned RFC value for draft-ietf-netconf-keystore

* "DDDD" --> the assigned RFC value for draft-ietf-netconf-tcp-client-server

* "FFFF" --> the assigned RFC value for this draft

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

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

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

* Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines a grouping for a generic TLS client, the second defines a grouping for a generic TLS server, and the third defines identities and groupings common to both the client and the server (TLS is defined in [RFC5246]). It is intended that these groupings will be used by applications using the TLS protocol. For instance, these groupings could be used to help define the data model for an HTTPS [RFC2818] server or a NETCONF over TLS [RFC7589] based server.

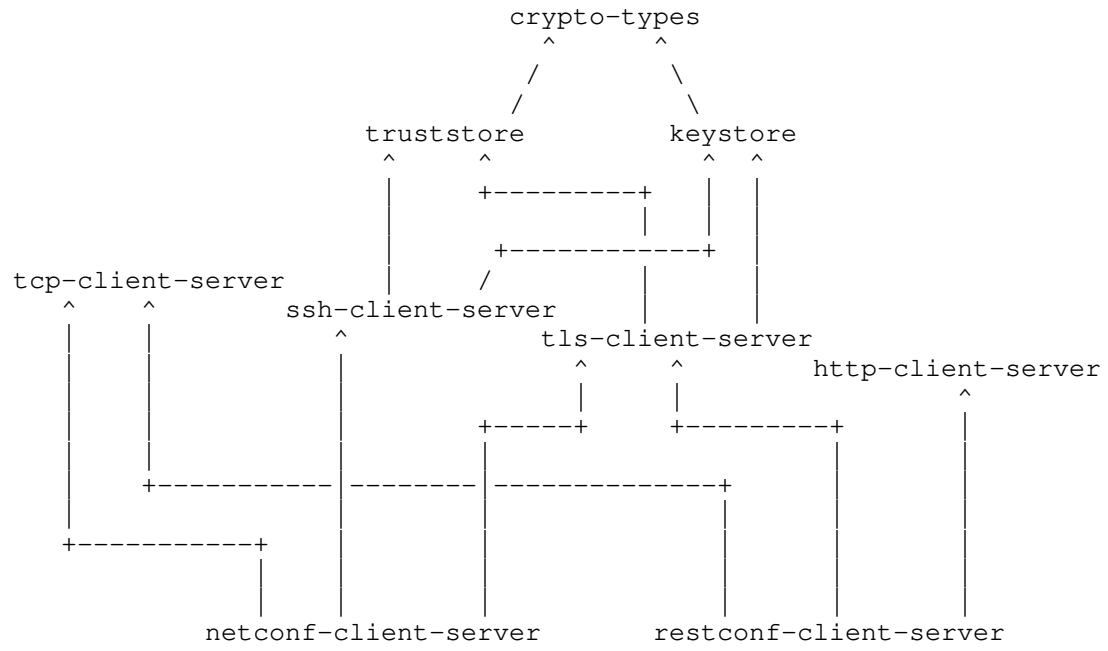
The client and server YANG modules in this document each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "ssh-server-grouping" grouping for the TLS parts it provides, while adding data nodes for the TCP-level call-home configuration.

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

2. The "ietf-tls-common" Module

The TLS common model presented in this section contains identities and groupings common to both TLS clients and TLS servers. The "hello-params-grouping" grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for TLS clients and TLS servers that are capable of doing so and may serve to make TLS clients and TLS servers compliant with local security policies. This model supports both TLS1.2 [RFC5246] and TLS 1.3 [RFC8446].

TLS 1.2 and TLS 1.3 have different ways defining their own supported cryptographic algorithms, see TLS and DTLS IANA registries page (https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml):

* TLS 1.2 defines four categories of registries for cryptographic algorithms: TLS Cipher Suites, TLS SignatureAlgorithm, TLS HashAlgorithm, TLS Supported Groups. TLS Cipher Suites plays the role of combining all of them into one set, as each value of the set represents a unique and feasible combination of all the cryptographic algorithms, and thus the other three registry categories do not need to be considered here. In this document, the TLS common model only chooses those TLS1.2 algorithms in TLS Cipher Suites which are marked as recommended: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256, TLS_DHE_RSA_WITH_AES_256_GCM_SHA384, TLS_DHE_PSK_WITH_AES_128_GCM_SHA256, TLS_DHE_PSK_WITH_AES_256_GCM_SHA384, and so on. All chosen algorithms are enumerated in Table 1-1 below;

* TLS 1.3 defines its supported algorithms differently. Firstly, it defines three categories of registries for cryptographic algorithms: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Secondly, all three of these categories are useful, since they represent different parts of all the supported algorithms respectively. Thus, all of these registries categories are considered here. In this draft, the TLS common model chooses only those TLS1.3 algorithms specified in B.4, 4.2.3, 4.2.7 of [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the "hello-params-grouping" grouping should include three parameters for configuring its permitted TLS algorithms, which are: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Note that TLS1.2 only uses TLS Cipher Suites.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive.

2.1. Data Model Overview

2.1.1. Features

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

```
   Features:
     +-- tls-1_0
     +-- tls-1_1
     +-- tls-1_2
     +-- tls-1_3
     +-- tls-ecc
     +-- tls-dhe
     +-- tls-3des
     +-- tls-gcm
     +-- tls-sha2
```

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-tls-common" module:

```
   Identities:
     +-- tls-version-base
     |  +-- tls-1.0
     |  +-- tls-1.1
     |  +-- tls-1.2
     +-- cipher-suite-base
        +-- rsa-with-aes-128-cbc-sha
        +-- rsa-with-aes-256-cbc-sha
        +-- rsa-with-aes-128-cbc-sha256
        +-- rsa-with-aes-256-cbc-sha256
        +-- dhe-rsa-with-aes-128-cbc-sha
        +-- dhe-rsa-with-aes-256-cbc-sha
        +-- dhe-rsa-with-aes-128-cbc-sha256
        +-- dhe-rsa-with-aes-256-cbc-sha256
        +-- ecdhe-ecdsa-with-aes-128-cbc-sha256
        +-- ecdhe-ecdsa-with-aes-256-cbc-sha384
        +-- ecdhe-rsa-with-aes-128-cbc-sha256
        +-- ecdhe-rsa-with-aes-256-cbc-sha384
        +-- ecdhe-ecdsa-with-aes-128-gcm-sha256
        +-- ecdhe-ecdsa-with-aes-256-gcm-sha384
        +-- ecdhe-rsa-with-aes-128-gcm-sha256
        +-- ecdhe-rsa-with-aes-256-gcm-sha384
        +-- rsa-with-3des-ede-cbc-sha
        +-- ecdhe-rsa-with-3des-ede-cbc-sha
        +-- ecdhe-rsa-with-aes-128-cbc-sha
        +-- ecdhe-rsa-with-aes-256-cbc-sha
```

Comments:

* The diagram shows that there are two base identities.
* One base identity is used to specific TLS versions, while the other is used to specify cipher-suites.

* These base identities are "abstract", in the object orientied programming sense, in that they only define a "class" of things, rather than a specific thing.

2.1.3. Groupings

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

```
Groupings:
  +-- hello-params-grouping
```

Each of these groupings are presented in the following subsections.

2.1.3.1. The "hello-params-grouping" Grouping

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

```
  grouping hello-params-grouping
    +-- tls-versions
    |  +-- tls-version*   identityref
    +-- cipher-suites
       +-- cipher-suite*   identityref
```

Comments:

* This grouping is used by both the "tls-client-grouping" and the "tls-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.

* This grouping enables client and server configurations to specify the TLS versions and cipher suites that are to be used when establishing TLS sessions.

* The "cipher-suites" list is "ordered-by user".

2.1.4. Protocol-accessible Nodes

The "ietf-tls-common" module does not contain any protocol-accessible nodes, but the module needs to be "implemented", as described in Section 5.6.5 of [RFC7950], in order for the identities in Section 2.1.2 to be defined.

2.2. Example Usage

This section shows how it would appear if the "hello-params-grouping" grouping were populated with some data.

```
   <hello-params
      xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common"
      xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
     <tls-versions>
       <tls-version>tlscmn:tls-1.1</tls-version>
       <tls-version>tlscmn:tls-1.2</tls-version>
     </tls-versions>
     <cipher-suites>
       <cipher-suite>tlscmn:dhe-rsa-with-aes-128-cbc-sha</cipher-suite>
       <cipher-suite>tlscmn:rsa-with-aes-128-cbc-sha</cipher-suite>
       <cipher-suite>tlscmn:rsa-with-3des-ede-cbc-sha</cipher-suite>
     </cipher-suites>
   </hello-params>
```

2.3. YANG Module

This YANG module has a normative references to [RFC4346], [RFC5246], [RFC5288], [RFC5289], and [RFC8422].

This YANG module has a informative references to [RFC2246], [RFC4346], [RFC5246], and [RFC8446].

```
   <CODE BEGINS> file "ietf-tls-common@2020-07-10.yang"

   module ietf-tls-common {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
     prefix tlscmn;

     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 <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

      description
       "This module defines a common features, identities, and
        groupings for Transport Layer Security (TLS).

        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 FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
        itself for full legal notices.

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

    revision 2020-07-10 {
      description
        "Initial version";
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    // Features

    feature tls-1_0 {
      description
        "TLS Protocol Version 1.0 is supported.";
      reference
        "RFC 2246: The TLS Protocol Version 1.0";
    }

    feature tls-1_1 {
      description
        "TLS Protocol Version 1.1 is supported.";
      reference
        "RFC 4346: The Transport Layer Security (TLS) Protocol
                   Version 1.1";
    }

    feature tls-1_2 {
      description
        "TLS Protocol Version 1.2 is supported.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    feature tls-1_3 {
```

```
      description
        "TLS Protocol Version 1.2 is supported.";
      reference
        "RFC 8446: The Transport Layer Security (TLS) Protocol
                   Version 1.3";
    }

    feature tls-ecc {
      description
        "Elliptic Curve Cryptography (ECC) is supported for TLS.";
      reference
        "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                   for Transport Layer Security (TLS)";
    }

    feature tls-dhe {
      description
        "Ephemeral Diffie-Hellman key exchange is supported for TLS.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    feature tls-3des {
      description
        "The Triple-DES block cipher is supported for TLS.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    feature tls-gcm {
      description
        "The Galois/Counter Mode authenticated encryption mode is
         supported for TLS.";
      reference
        "RFC 5288: AES Galois Counter Mode (GCM) Cipher Suites for
                   TLS";
    }

    feature tls-sha2 {
      description
        "The SHA2 family of cryptographic hash functions is supported
         for TLS.";
      reference
        "FIPS PUB 180-4: Secure Hash Standard (SHS)";
    }
```

```
     // Identities

     identity tls-version-base {
       description
         "Base identity used to identify TLS protocol versions.";
     }

     identity tls-1.0 {
       if-feature "tls-1_0";
       base tls-version-base;
       description
         "TLS Protocol Version 1.0.";
       reference
         "RFC 2246: The TLS Protocol Version 1.0";
     }

     identity tls-1.1 {
       if-feature "tls-1_1";
       base tls-version-base;
       description
         "TLS Protocol Version 1.1.";
       reference
         "RFC 4346: The Transport Layer Security (TLS) Protocol
                    Version 1.1";
     }

     identity tls-1.2 {
       if-feature "tls-1_2";
       base tls-version-base;
       description
         "TLS Protocol Version 1.2.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity cipher-suite-base {
       description
         "Base identity used to identify TLS cipher suites.";
     }

     identity rsa-with-aes-128-cbc-sha {
       base cipher-suite-base;
       description
         "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
```

```
    }

    identity rsa-with-aes-256-cbc-sha {
      base cipher-suite-base;
      description
        "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    identity rsa-with-aes-128-cbc-sha256 {
      if-feature "tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA256.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    identity rsa-with-aes-256-cbc-sha256 {
      if-feature "tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA256.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    identity dhe-rsa-with-aes-128-cbc-sha {
      if-feature "tls-dhe";
      base cipher-suite-base;
      description
        "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    identity dhe-rsa-with-aes-256-cbc-sha {
      if-feature "tls-dhe";
      base cipher-suite-base;
      description
        "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
```

```
                Version 1.2";
    }

    identity dhe-rsa-with-aes-128-cbc-sha256 {
      if-feature "tls-dhe and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA256.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                Version 1.2";
    }

    identity dhe-rsa-with-aes-256-cbc-sha256 {
      if-feature "tls-dhe and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA256.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                Version 1.2";
    }

    identity ecdhe-ecdsa-with-aes-128-cbc-sha256 {
      if-feature "tls-ecc and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256.";
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity ecdhe-ecdsa-with-aes-256-cbc-sha384 {
      if-feature "tls-ecc and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384.";
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity ecdhe-rsa-with-aes-128-cbc-sha256 {
      if-feature "tls-ecc and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256.";
```

```
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                   SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity ecdhe-rsa-with-aes-256-cbc-sha384 {
      if-feature "tls-ecc and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384.";
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                   SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity ecdhe-ecdsa-with-aes-128-gcm-sha256 {
      if-feature "tls-ecc and tls-gcm and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.";
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                   SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity ecdhe-ecdsa-with-aes-256-gcm-sha384 {
      if-feature "tls-ecc and tls-gcm and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.";
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                   SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity ecdhe-rsa-with-aes-128-gcm-sha256 {
      if-feature "tls-ecc and tls-gcm and tls-sha2";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.";
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                   SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity ecdhe-rsa-with-aes-256-gcm-sha384 {
      if-feature "tls-ecc and tls-gcm and tls-sha2";
      base cipher-suite-base;
```

```
      description
        "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.";
      reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
                   SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity rsa-with-3des-ede-cbc-sha {
      if-feature "tls-3des";
      base cipher-suite-base;
      description
        "Cipher suite TLS_RSA_WITH_3DES_EDE_CBC_SHA.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    identity ecdhe-rsa-with-3des-ede-cbc-sha {
      if-feature "tls-ecc and tls-3des";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA.";
      reference
        "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                   for Transport Layer Security (TLS)";
    }

    identity ecdhe-rsa-with-aes-128-cbc-sha {
      if-feature "tls-ecc";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA.";
      reference
        "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                   for Transport Layer Security (TLS)";
    }

    identity ecdhe-rsa-with-aes-256-cbc-sha {
      if-feature "tls-ecc";
      base cipher-suite-base;
      description
        "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA.";
      reference
        "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                   for Transport Layer Security (TLS)";
    }

    // Groupings
```

```
    grouping hello-params-grouping {
      description
        "A reusable grouping for TLS hello message parameters.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
      container tls-versions {
        description
          "Parameters regarding TLS versions.";
        leaf-list tls-version {
          type identityref {
            base tls-version-base;
          }
          description
            "Acceptable TLS protocol versions.

             If this leaf-list is not configured (has zero elements)
             the acceptable TLS protocol versions are implementation-
             defined.";
        }
      }
      container cipher-suites {
        description
          "Parameters regarding cipher suites.";
        leaf-list cipher-suite {
          type identityref {
            base cipher-suite-base;
          }
          ordered-by user;
          description
            "Acceptable cipher suites in order of descending
             preference.  The configured host key algorithms should
             be compatible with the algorithm used by the configured
             private key.  Please see Section 5 of RFC FFFF for
             valid combinations.

             If this leaf-list is not configured (has zero elements)
             the acceptable cipher suites are implementation-
             defined.";
          reference
            "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
        }
      }
    }
  }

  <CODE ENDS>
```

3. The "ietf-tls-client" Module

3.1. Data Model Overview

3.1.1. Features

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

```
Features:
  +-- tls-client-hello-params-config
  +-- tls-client-keepalives
  +-- x509-certificate-auth
  +-- raw-public-key-auth
  +-- psk-auth
```

3.1.2. Groupings

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

```
Groupings:
  +-- tls-client-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "tls-client-grouping" Grouping

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

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

     grouping tls-client-grouping
       +-- client-identity!
       |  +-- (auth-type)
       |     +--:(certificate) {x509-certificate-auth}?
       |     |  +-- certificate
       |     |     +---u ks:local-or-keystore-end-entity-cert-with-key-\
   grouping
       |     +--:(raw-public-key) {raw-public-key-auth}?
       |     |  +-- raw-private-key
       |     |     +---u ks:local-or-keystore-asymmetric-key-grouping
       |     +--:(psk) {psk-auth}?
       |        +-- psk
       |           +---u ks:local-or-keystore-symmetric-key-grouping
       +-- server-authentication
       |  +-- ca-certs! {x509-certificate-auth}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- ee-certs! {x509-certificate-auth}?
       |  |  +---u ts:local-or-truststore-certs-grouping
       |  +-- raw-public-keys! {raw-public-key-auth}?
       |  |  +---u ts:local-or-truststore-public-keys-grouping
       |  +-- psks! {psk-auth}?
       +-- hello-params {tls-client-hello-params-config}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives {tls-client-keepalives}?
          +-- peer-allowed-to-send?   empty
          +-- test-peer-aliveness!
             +-- max-wait?       uint16
             +-- max-attempts?   uint8
```

Comments:

* The "client-identity" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures identity credentials, each enabled by a "feature" statement defined in Section 3.1.1.

* The "server-authentication" node configures trust anchors for authenticating the TLS server, with each option enabled by a "feature" statement.

* The "hello-params" node , which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the TLS server. The aliveness-test occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-tls-client" module does not contain any protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "tls-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the client identity and server authentication:

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

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

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <certificate>
```

```
        <local-definition>
          <public-key-format>ct:subject-public-key-info-format</public\
   -key-format>
          <public-key>base64encodedvalue==</public-key>
          <private-key-format>ct:rsa-private-key-format</private-key-f\
   ormat>
          <cleartext-private-key>base64encodedvalue==</cleartext-priva\
   te-key>
          <cert-data>base64encodedvalue==</cert-data>
        </local-definition>
      </certificate>
      <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A TIME
      <raw-private-key>
        <local-definition>
          <public-key-format>ct:subject-public-key-info-format</public\
   -key-format>
          <public-key>base64encodedvalue==</public-key>
          <private-key-format>ct:rsa-private-key-format</private-key-f\
   ormat>
          <cleartext-private-key>base64encodedvalue==</cleartext-priva\
   te-key>
        </local-definition>
      </raw-private-key>
      <psk>
        <local-definition>
          <key-format>ct:octet-string-key-format</key-format>
          <cleartext-key>base64encodedvalue==</cleartext-key>
        </local-definition>
      </psk>
      -->
    </client-identity>

    <!-- which certificates will this client trust -->
    <server-authentication>
      <ca-certs>
        <local-definition>
          <certificate>
            <name>Server Cert Issuer #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Server Cert Issuer #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </local-definition>
      </ca-certs>
      <ee-certs>
        <local-definition>
```

```
          <certificate>
            <name>My Application #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>My Application #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </local-definition>
      </ee-certs>
      <raw-public-keys>
        <local-definition>
          <public-key>
            <name>corp-fw1</name>
            <public-key-format>ct:subject-public-key-info-format</publ\
ic-key-format>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
          <public-key>
            <name>corp-fw1</name>
            <public-key-format>ct:subject-public-key-info-format</publ\
ic-key-format>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
        </local-definition>
      </raw-public-keys>
      <psks/>
    </server-authentication>

    <keepalives>
      <test-peer-aliveness>
        <max-wait>30</max-wait>
        <max-attempts>3</max-attempts>
      </test-peer-aliveness>
    </keepalives>

  </tls-client>
```

The following configuration example uses keystore-references for the client identity and truststore-references for server authentication: from the keystore:

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

  <tls-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client">

    <!-- how this client will authenticate itself to the server -->
    <client-identity>
```

```
        <certificate>
          <keystore-reference>
            <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
            <certificate>ex-rsa-cert</certificate>
          </keystore-reference>
        </certificate>
        <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A TIME
        <raw-private-key>
          <keystore-reference>raw-private-key</keystore-reference>
        </raw-private-key>
        <psk>
          <keystore-reference>encrypted-symmetric-key</keystore-referenc\
    e>
        </psk>
        -->
      </client-identity>

      <!-- which certificates will this client trust -->
      <server-authentication>
        <ca-certs>
          <truststore-reference>trusted-server-ca-certs</truststore-refe\
    rence>
        </ca-certs>
        <ee-certs>
          <truststore-reference>trusted-server-ee-certs</truststore-refe\
    rence>
        </ee-certs>
        <raw-public-keys>
          <truststore-reference>Raw Public Keys for TLS Servers</truststt\
    ore-reference>
        </raw-public-keys>
        <psks/>
      </server-authentication>

      <keepalives>
        <test-peer-aliveness>
          <max-wait>30</max-wait>
          <max-attempts>3</max-attempts>
        </test-peer-aliveness>
      </keepalives>

    </tls-client>
```

3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-tls-client@2020-07-10.yang"

module ietf-tls-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix tlsc;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2020-07-10; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
     Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
```

```
      "This module defines reusable groupings for TLS clients that
       can be used as a basis for specific TLS client instances.

       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 FFFF
       (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
       itself for full legal notices.

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

    revision 2020-07-10 {
      description
        "Initial version";
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    // Features

    feature tls-client-hello-params-config {
      description
        "TLS hello message parameters are configurable on a TLS
         client.";
    }

    feature tls-client-keepalives {
      description
        "Per socket TLS keepalive parameters are configurable for
         TLS clients on the server implementing this feature.";
    }

    feature x509-certificate-auth {
      description
        "Indicates that the client supports authenticating servers
```

```
          using X.509 certificates.";
      }

      feature raw-public-key-auth {
        description
          "Indicates that the client supports authenticating servers
           using ray public keys.";
      }

      feature psk-auth {
        description
          "Indicates that the client supports authenticating servers
           using PSKs (pre-shared or pairwise-symmetric keys).";
      }



      // Groupings

      grouping tls-client-grouping {
        description
          "A reusable grouping for configuring a TLS client without
           any consideration for how an underlying TCP session is
           established.

           Note that this grouping uses fairly typical descendent
           node names such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue (e.g., by wrapping
           the 'uses' statement in a container called
           'tls-client-parameters').  This model purposely does
           not do this itself so as to provide maximum flexibility
           to consuming models.";

        container client-identity {
          nacm:default-deny-write;
          presence
            "Indicates that TLS-level client authentication
             is sent.  Present so that the 'choice' node's
             mandatory true doesn't imply that a client
             identity must be configured.";
          description
            "Identity credentials the TLS client MAY present when
             establishing a connection to a TLS server.  If not
             configured, then client authentication is presumed to
             occur a protocol layer above TLS.  When configured,
             and requested by the TLS server when establishing a
             TLS session, these credentials are passed in the
```

```
           Certificate message defined in Section 7.4.2 of
           RFC 5246.";
         reference
           "RFC 5246: The Transport Layer Security (TLS) Protocol
                      Version 1.2
           RFC CCCC: A YANG Data Model for a Keystore";
         choice auth-type {
           mandatory true;
           description
             "A choice amongst available authentication types.";
             /* FIXME: delete now?
              Not mandatory as client identity MAY be provided
              by another layer in the protocol stack (e.g., an
              HTTP authentication mechanism).";*/
           case certificate {
             if-feature x509-certificate-auth;
             container certificate {
               description
                 "Specifies the client identity using a certificate.";
               uses
                 ks:local-or-keystore-end-entity-cert-with-key-grouping{
                 refine "local-or-keystore/local/local-definition" {
                   must 'public-key-format'
                        + ' = "ct:subject-public-key-info-format"';
                 }
                 refine "local-or-keystore/keystore/keystore-reference"
                        + "/asymmetric-key" {
                   must 'deref(.)/../ks:public-key-format'
                        + ' = "ct:subject-public-key-info-format"';
                 }
               }
             }
           }
           case raw-public-key {
             if-feature raw-public-key-auth;
             container raw-private-key {
               description
                 "Specifies the client identity using a raw
                  private key.";
               uses ks:local-or-keystore-asymmetric-key-grouping {
                 refine "local-or-keystore/local/local-definition" {
                   must 'public-key-format'
                        + ' = "ct:subject-public-key-info-format"';
                 }
                 refine "local-or-keystore/keystore"
                        + "/keystore-reference" {
                   must 'deref(.)/../ks:public-key-format'
                        + ' = "ct:subject-public-key-info-format"';
```

```
                }
              }
            }
          }
          case psk {
            if-feature psk-auth;
            container psk {
              description
                "Specifies the client identity using a PSK (pre-shared
                or pairwise-symmetric key).  Note that, when the PSK is
                configured as a Keystore reference, the key's 'name'
                node MAY be used as the PSK's ID when used by the TLS
                protocol.";
              uses ks:local-or-keystore-symmetric-key-grouping {
                augment "local-or-keystore/local/local-definition" {
                  if-feature "ks:local-definitions-supported";
                  description
                    "Adds an 'id' value when the PSK is used by TLS.";
                  leaf id {
                    type string;  // FIXME: is this the right type?
                    description
                      "The key id used in the TLS protocol for PSKs.";
                  }
                }
              }
            }
          }
        }
      } // container client-identity

      container server-authentication {
        nacm:default-deny-write;
        must 'ca-certs or ee-certs or raw-public-keys or psks';
        description
          "Specifies how the TLS client can authenticate TLS servers.
           Any combination of credentials is additive and unordered.

           Note that no configuration is required for PSK (pre-shared
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../client-
           identity' node.";
        container ca-certs {
          if-feature "x509-certificate-auth";
          presence
            "Indicates that the TLS client can authenticate TLS servers
             using configured certificate authority certificates.";
          description
            "A set of certificate authority (CA) certificates used by
```

```
            the TLS client to authenticate TLS server certificates.
            A server certificate is authenticated if it has a valid
            chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs { // FIXME: plural too much?
          if-feature "x509-certificate-auth";
          presence
            "Indicates that the TLS client can authenticate TLS
             servers using configured server certificates.";
          description
            "A set of server certificates (i.e., end entity
             certificates) used by the TLS client to authenticate
             certificates presented by TLS servers.  A server
             certificate is authenticated if it is an exact
             match to a configured server certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container raw-public-keys {
          if-feature "raw-public-key-auth";
          presence
            "Indicates that the TLS client can authenticate TLS
             servers using configured server certificates.";
          description
            "A set of raw public keys used by the TLS client to
             authenticate raw public keys presented by the TLS
             server.  A raw public key is authenticated if it
             is an exact match to a configured raw public key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine "local-or-truststore/local/local-definition"
                   + "/public-key" {
              must 'public-key-format'
                   + ' = "ct:subject-public-key-info-format"';
            }
            refine "local-or-truststore/truststore"
                   + "/truststore-reference" {
              must 'deref(.)/../*/ts:public-key-format'
                   + ' = "ct:subject-public-key-info-format"';
            }
          }
        }
        container psks {
```

```
          if-feature "psk-auth";
          presence
            "Indicates that the TLS client can authenticate TLS servers
             using a configure PSK (pre-shared or pairwise-symmetric
             key).";
          description
            "No configuration is required since the PSK value is the
             same as PSK value configured in the 'client-identity'
             node.";
        }
      } // container server-authentication

      container hello-params {
        nacm:default-deny-write;
        if-feature "tls-client-hello-params-config";
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      } // container hello-params

      container keepalives {
        nacm:default-deny-write;
        if-feature "tls-client-keepalives";
        description
          "Configures the keepalive policy for the TLS client.";
        leaf peer-allowed-to-send {
          type empty;
          description
            "Indicates that the remote TLS server is allowed to send
             HeartbeatRequest messages, as defined by RFC 6520
             to this TLS client.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
        }
        container test-peer-aliveness {
          presence
            "Indicates that the TLS client proactively tests the
             aliveness of the remote TLS server.";
          description
            "Configures the keep-alive policy to proactively test
             the aliveness of the TLS server.  An unresponsive
             TLS server is dropped after approximately max-wait
             * max-attempts seconds.  The TLS client MUST send
             HeartbeatRequest messages, as defined by RFC 6520.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
```

```
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "30";
            description
              "Sets the amount of time in seconds after which if
               no data has been received from the TLS server, a
               TLS-level message will be sent to test the
               aliveness of the TLS server.";
          }
          leaf max-attempts {
            type uint8;
            default "3";
            description
              "Sets the maximum number of sequential keep-alive
               messages that can fail to obtain a response from
               the TLS server before assuming the TLS server is
               no longer alive.";
          }
        }
      }
    } // grouping tls-client-grouping
  } // module ietf-tls-client

  <CODE ENDS>
```

4. The "ietf-tls-server" Module

4.1. Data Model Overview

4.1.1. Features

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

```
Features:
  +-- tls-server-hello-params-config
  +-- tls-server-keepalives
  +-- client-auth-config-supported
  +-- x509-certificate-auth
  +-- raw-public-key-auth
  +-- psk-auth
```

4.1.2. Groupings

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

```
Groupings:
  +-- tls-server-grouping
```

Each of these groupings are presented in the following subsections.

4.1.2.1. The "tls-server-grouping" Grouping

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

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

  grouping tls-server-grouping
    +-- server-identity
    |  +-- (auth-type)
    |     +--:(certificate) {x509-certificate-auth}?
    |     |  +-- certificate
    |     |     +---u ks:local-or-keystore-end-entity-cert-with-key-\
grouping
    |     +--:(raw-private-key) {raw-public-key-auth}?
    |     |  +-- raw-private-key
    |     |     +---u ks:local-or-keystore-asymmetric-key-grouping
    |     +--:(psk) {psk-auth}?
    |        +-- psk
    |           +---u ks:local-or-keystore-symmetric-key-grouping
    +-- client-authentication! {client-auth-config-supported}?
    |  +-- ca-certs! {x509-certificate-auth}?
    |  |  +---u ts:local-or-truststore-certs-grouping
    |  +-- ee-certs! {x509-certificate-auth}?
    |  |  +---u ts:local-or-truststore-certs-grouping
    |  +-- raw-public-keys! {raw-public-key-auth}?
    |  |  +---u ts:local-or-truststore-public-keys-grouping
    |  +-- psks! {psk-auth}?
    +-- hello-params {tls-server-hello-params-config}?
    |  +---u tlscmn:hello-params-grouping
    +-- keepalives {tls-server-keepalives}?
       +-- peer-allowed-to-send?   empty
       +-- test-peer-aliveness!
          +-- max-wait?       uint16
          +-- max-attempts?   uint8
```

Comments:

* The "server-identity" node configures identity credentials, each of which is enabled by a "feature".

* The "client-authentication" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures trust anchors for authenticating the TLS client, with each option enabled by a "feature" statement.

* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a flag enabling the TLS client to test the aliveness of the TLS server, as well as a "presence" container for testing the aliveness of the TLSi client. The aliveness-tests occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "local-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "local-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.2 of [I-D.ietf-netconf-trust-anchors].
  - The "local-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

4.1.3. Protocol-accessible Nodes

The "ietf-tls-server" module does not contain any protocol-accessible nodes.

4.2. Example Usage

This section presents two examples showing the "tls-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following configuration example uses local-definitions for the server identity and client authentication:

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

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

  <!-- how this server will authenticate itself to the client -->
  <server-identity>
    <certificate>
      <local-definition>
        <public-key-format>ct:subject-public-key-info-format</public\
-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-f\
ormat>
        <cleartext-private-key>base64encodedvalue==</cleartext-priva\
te-key>
        <cert-data>base64encodedvalue==</cert-data>
      </local-definition>
    </certificate>
    <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A TIME
    <raw-private-key>
      <local-definition>
        <public-key-format>ct:subject-public-key-info-format</public\
-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-f\
ormat>
        <cleartext-private-key>base64encodedvalue==</cleartext-priva\
te-key>
      </local-definition>
    </raw-private-key>
    <psk>
      <local-definition>
        <key-format>ct:octet-string-key-format</key-format>
```

```
          <cleartext-key>base64encodedvalue==</cleartext-key>
        </local-definition>
      </psk>
      -->
    </server-identity>

    <!-- which certificates will this server trust -->
    <client-authentication>
      <ca-certs>
        <local-definition>
          <certificate>
            <name>Identity Cert Issuer #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Identity Cert Issuer #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </local-definition>
      </ca-certs>
      <ee-certs>
        <local-definition>
          <certificate>
            <name>Application #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Application #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </local-definition>
      </ee-certs>
      <raw-public-keys>
        <local-definition>
          <public-key>
            <name>User A</name>
            <public-key-format>ct:subject-public-key-info-format</publ\
  ic-key-format>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
          <public-key>
            <name>User B</name>
            <public-key-format>ct:subject-public-key-info-format</publ\
  ic-key-format>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
        </local-definition>
      </raw-public-keys>
```

```
        <psks/>
      </client-authentication>

      <keepalives>
        <peer-allowed-to-send/>
      </keepalives>

    </tls-server>
```

The following configuration example uses keystore-references for the server identity and truststore-references for client authentication: from the keystore:

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

   <tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">

     <!-- how this server will authenticate itself to the client -->
     <server-identity>
       <certificate>
         <keystore-reference>
           <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
           <certificate>ex-rsa-cert</certificate>
         </keystore-reference>
       </certificate>
       <!-- TESTED, BUT COMMENTED OUT DUE TO ONLY ONE ALLOWED AT A TIME
       <raw-private-key>
         <keystore-reference>raw-private-key</keystore-reference>
       </raw-private-key>
       <psk>
         <keystore-reference>encrypted-symmetric-key</keystore-referenc\
   e>
       </psk>
       -->
     </server-identity>

     <!-- which certificates will this server trust -->
     <client-authentication>
       <ca-certs>
         <truststore-reference>trusted-client-ca-certs</truststore-refe\
   rence>
       </ca-certs>
       <ee-certs>
         <truststore-reference>trusted-client-ee-certs</truststore-refe\
   rence>
       </ee-certs>
       <raw-public-keys>
         <truststore-reference>Raw Public Keys for TLS Clients</truststt\
   ore-reference>
       </raw-public-keys>
       <psks/>
     </client-authentication>

     <keepalives>
       <peer-allowed-to-send/>
     </keepalives>

   </tls-server>
```

4.3. YANG Module

This YANG module has a normative references to [RFC5246], [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore].

<CODE BEGINS> file "ietf-tls-server@2020-07-10.yang"

```
module ietf-tls-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix tlss;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2020-07-10; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  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 <mailto:kent+ietf@watsen.net>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

      description
        "This module defines reusable groupings for TLS servers that
         can be used as a basis for specific TLS server instances.

         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 FFFF
         (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
         itself for full legal notices.

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

      revision 2020-07-10 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Features

      feature tls-server-hello-params-config {
        description
          "TLS hello message parameters are configurable on a TLS
           server.";
      }

      feature tls-server-keepalives {
        description
          "Per socket TLS keepalive parameters are configurable for
           TLS servers on the server implementing this feature.";
```

```
  }

  feature client-auth-config-supported {
    description
      "Indicates that the configuration for how to authenticate
       clients can be configured herein, as opposed to in an
       application specific location.  That is, to support the
       consuming data models that prefer to place client
       authentication with client definitions, rather then
       in a data model principally concerned with configuring
       the transport.";
  }

  feature x509-certificate-auth {
    description
      "Indicates that the server supports authenticating clients
       using X.509 certificates.";
  }

  feature raw-public-key-auth {
    description
      "Indicates that the server supports authenticating clients
       using ray public keys.";
  }

  feature psk-auth {
    description
      "Indicates that the server supports authenticating clients
       using PSKs (pre-shared or pairwise-symmetric keys).";
  }



  // Groupings

  grouping tls-server-grouping {
    description
      "A reusable grouping for configuring a TLS server without
       any consideration for how underlying TCP sessions are
       established.

       Note that this grouping uses fairly typical descendent
       node names such that a stack of 'uses' statements will
       have name conflicts.  It is intended that the consuming
       data model will resolve the issue (e.g., by wrapping
       the 'uses' statement in a container called
       'tls-server-parameters').  This model purposely does
       not do this itself so as to provide maximum flexibility
```

```
          to consuming models.";

        container server-identity {
          nacm:default-deny-write;
          description
            "A locally-defined or referenced end-entity certificate,
             including any configured intermediate certificates, the
             TLS server will present when establishing a TLS connection
             in its Certificate message, as defined in Section 7.4.2
             in RFC 5246.";
          reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
                       Version 1.2
             RFC CCCC: A YANG Data Model for a Keystore";
          choice auth-type {
            mandatory true;
            description
              "A choice amongst authentication types.";
            case certificate {
              if-feature x509-certificate-auth;
              container certificate {
                description
                  "Specifies the server identity using a certificate.";
                uses
                  ks:local-or-keystore-end-entity-cert-with-key-grouping{
                  refine "local-or-keystore/local/local-definition" {
                    must 'public-key-format'
                         + ' = "ct:subject-public-key-info-format"';
                  }
                  refine "local-or-keystore/keystore/keystore-reference"
                         + "/asymmetric-key" {
                    must 'deref(.)/../ks:public-key-format'
                         + ' = "ct:subject-public-key-info-format"';
                  }
                }
              }
            }
            case raw-private-key {
              if-feature raw-public-key-auth;
              container raw-private-key {
                description
                  "Specifies the server identity using a raw
                   private key.";
                uses ks:local-or-keystore-asymmetric-key-grouping {
                  refine "local-or-keystore/local/local-definition" {
                    must 'public-key-format'
                         + ' = "ct:subject-public-key-info-format"';
```

```
                }
                refine "local-or-keystore/keystore/keystore-reference"{
                  must 'deref(.)/../ks:public-key-format'
                       + ' = "ct:subject-public-key-info-format"';
                }
              }
            }
          }
          case psk {
            if-feature psk-auth;
            container psk {
              description
                "Specifies the server identity using a PSK (pre-shared
                or pairwise-symmetric key).  Note that, when the PSK is
                configured as a Keystore reference, the key's 'name'
                node MAY be used as the PSK's ID when used by the TLS
                protocol.";
              uses ks:local-or-keystore-symmetric-key-grouping {
                augment "local-or-keystore/local/local-definition" {
                  if-feature "ks:local-definitions-supported";
                  description
                    "An 'id' value for when the PSK is used by TLS.";
                  leaf id {
                    type string;  // FIXME: is this the right type?
                    description
                      "The key id used in the TLS protocol for PSKs.";
                  }
                }
              }
            }
          }
        }
      } // container server-identity

      container client-authentication {
        if-feature "client-auth-config-supported";
        nacm:default-deny-write;
        must 'ca-certs or ee-certs or raw-public-keys or psks';
        presence
          "Indicates that client authentication is supported (i.e.,
           that the server will request clients send certificates).
           If not configured, the TLS server SHOULD NOT request the
           TLS clients provide authentication credentials.";
        description
          "Specifies how the TLS server can authenticate TLS clients.
           Any combination of credentials is additive and unordered.

           Note that no configuration is required for PSK (pre-shared
```

```
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../server-
           identity' node.";
        container ca-certs {
          if-feature "x509-certificate-auth";
          presence
            "Indicates that the TLS server can authenticate TLS clients
             using configured certificate authority certificates.";
          description
            "A set of certificate authority (CA) certificates used by
             the TLS server to authenticate TLS client certificates. A
             client certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container ee-certs {   // FIXME: plural too much?
          if-feature "x509-certificate-auth";
          presence
            "Indicates that the TLS server can authenticate TLS
             clients using configured client certificates.";
          description
            "A set of client certificates (i.e., end entity
             certificates) used by the TLS server to authenticate
             certificates presented by TLS clients. A client
             certificate is authenticated if it is an exact
             match to a configured client certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-certs-grouping;
        }
        container raw-public-keys {
          if-feature "raw-public-key-auth";
          presence
            "Indicates that the TLS server can authenticate TLS
             clients using raw public keys.";
          description
            "A set of raw public keys used by the TLS server to
             authenticate raw public keys presented by the TLS
             client.  A raw public key is authenticated if it
             is an exact match to a configured raw public key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:local-or-truststore-public-keys-grouping {
            refine "local-or-truststore/local/local-definition"
                   + "/public-key" {
              must 'public-key-format'
```

```
                + ' = "ct:subject-public-key-info-format"';
          }
          refine "local-or-truststore/truststore"
                + "/truststore-reference" {
            must 'deref(.)/../*/ts:public-key-format'
                + ' = "ct:subject-public-key-info-format"';
          }
        }
      }
      container psks {
        if-feature "psk-auth";
        presence
          "Indicates that the TLS server can authenticate the TLS
           client using its PSK (pre-shared or pairwise-symmetric
           key).";
        description
          "No configuration is required since the PSK value is the
           same as PSK value configured in the 'client-identity'
           node.";
      }
    } // container client-authentication

    container hello-params {
      nacm:default-deny-write;
      if-feature "tls-server-hello-params-config";
      uses tlscmn:hello-params-grouping;
      description
        "Configurable parameters for the TLS hello message.";
    } // container hello-params

    container keepalives {
      nacm:default-deny-write;
      if-feature "tls-server-keepalives";
      description
        "Configures the keepalive policy for the TLS server.";
      leaf peer-allowed-to-send {
        type empty;
        description
          "Indicates that the remote TLS client is allowed to send
           HeartbeatRequest messages, as defined by RFC 6520
           to this TLS server.";
        reference
          "RFC 6520: Transport Layer Security (TLS) and Datagram
           Transport Layer Security (DTLS) Heartbeat Extension";
      }
      container test-peer-aliveness {
        presence
          "Indicates that the TLS server proactively tests the
```

```
             aliveness of the remote TLS client.";
           description
             "Configures the keep-alive policy to proactively test
              the aliveness of the TLS client.  An unresponsive
              TLS client is dropped after approximately max-wait
              * max-attempts seconds.";
           leaf max-wait {
             type uint16 {
               range "1..max";
             }
             units "seconds";
             default "30";
             description
               "Sets the amount of time in seconds after which if
                no data has been received from the TLS client, a
                TLS-level message will be sent to test the
                aliveness of the TLS client.";
           }
           leaf max-attempts {
             type uint8;
             default "3";
             description
               "Sets the maximum number of sequential keep-alive
                messages that can fail to obtain a response from
                the TLS client before assuming the TLS client is
                no longer alive.";
           }
         }
       } // container keepalives
     } // grouping tls-server-grouping
   } // module ietf-tls-server

   <CODE ENDS>
```

5. Security Considerations

5.1. The "ietf-tls-common" YANG Module

The "ietf-tls-common" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. The "ietf-tls-client" YANG Module

The "ietf-tls-client" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All of the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.3. The "ietf-tls-server" YANG Module

The "ietf-tls-server" YANG module defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

> Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All of the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers three URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-tls-common
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-client
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-server
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

6.2. The "YANG Module Names" Registry

This document registers three YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-tls-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-common
prefix:       tlscmn
reference:    RFC FFFF

name:         ietf-tls-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-client
prefix:       tlsc
reference:    RFC FFFF

name:         ietf-tls-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-server
prefix:       tlss
reference:    RFC FFFF
```

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-

crypto-types-15, 20 May 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

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

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

[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>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <https://www.rfc-editor.org/info/rfc5289>.

[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>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[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>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[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>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

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

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

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

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <https://www.rfc-editor.org/info/rfc2246>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

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

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

Appendix A. Change Log

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

A.1. 00 to 01

* Noted that '0.0.0.0' and '::' might have special meanings.

* Renamed "keychain" to "keystore".

A.2. 01 to 02

* Removed the groupings containing transport-level configuration. Now modules contain only the transport-independent groupings.

* Filled in previously incomplete 'ietf-tls-client' module.

* Added cipher suites for various algorithms into new 'ietf-tls-common' module.

A.3. 02 to 03

* Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

* Fixed description statement for leaf 'trusted-ca-certs'.

A.4. 03 to 04

* Updated title to "YANG Groupings for TLS Clients and TLS Servers"

* Updated leafref paths to point to new keystore path

* Changed the YANG prefix for ietf-tls-common from 'tlscom' to 'tlscmn'.

* Added TLS protocol verions 1.0 and 1.1.

* Made author lists consistent

   * Now tree diagrams reference ietf-netmod-yang-tree-diagrams

   * Updated YANG to use typedefs around leafrefs to common keystore paths

   * Now inlines key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

   * Merged changes from co-author.

A.6. 05 to 06

   * Updated to use trust anchors from trust-anchors draft (was keystore draft)

   * Now Uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

A.7. 06 to 07

   * factored the tls-[client|server]-groupings into more reusable groupings.

   * added if-feature statements for the new "x509-certificates" feature defined in draft-ietf-netconf-trust-anchors.

A.8. 07 to 08

   * Added a number of compatibility matrices to Section 5 (thanks Frank!)

   * Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

A.9. 08 to 09

   * Updated examples to reflect update to groupings defined in the keystore draft.

   * Add TLS keepalives features and groupings.

   * Prefixed top-level TLS grouping nodes with 'tls-' and support mashups.

   * Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

* Reformatted the YANG modules.

A.11. 10 to 11

* Collapsed all the inner groupings into the top-level grouping.

* Added a top-level "demux container" inside the top-level grouping.

* Added NACM statements and updated the Security Considerations section.

* Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

A.12. 11 to 12

* In server model, made 'client-authentication' a 'presence' node indicating that the server supports client authentication.

* In the server model, added a 'required-or-optional' choice to 'client-authentication' to better support protocols such as RESTCONF.

* In the server model, added a 'local-or-external' choice to 'client-authentication' to better support consuming data models that prefer to keep client auth with client definitions than in a model principally concerned with the "transport".

* In both models, removed the "demux containers", floating the nacm:default-deny-write to each descendent node, and adding a note to model designers regarding the potential need to add their own demux containers.

* Fixed a couple references (section 2 --> section 3)

A.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

A.14. 12 to 13

* Removed 'container' under 'client-identity' to match server model.

* Updated examples to reflect change grouping in keystore module.

A.15. 13 to 14

* Removed the "certificate" container from "client-identity" in the ietf-tls-client module.

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

A.16. 14 to 15

* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:certificates-ref" to a container that uses "ts:local-or-truststore-certs-grouping".

A.17. 15 to 16

* Removed unnecessary if-feature statements in the -client and -server modules.

* Cleaned up some description statements in the -client and -server modules.

* Fixed a canonical ordering issue in ietf-tls-common detected by new pyang.

A.18. 16 to 17

* Removed choice local-or-external by removing the 'external' case and flattening the 'local' case and adding a "client-auth-config-supported" feature.

* Removed choice required-or-optional.

* Updated examples to include the "*-key-format" nodes.

* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:ssh-public-key-format" (must expr for ref'ed keys are TBD).

A.19. 17 to 18

* Removed the unused "external-client-auth-supported" feature.
* Made client-indentity optional, as there may be over-the-top auth instead.
* Added augment to uses of local-or-keystore-symmetric-key-grouping for a psk "id" node.
* Added missing presence container "psks" to ietf-tls-server's "client-authentication" container.
* Updated examples to reflect new "bag" addition to truststore.
* Removed feature-limited caseless 'case' statements to improve tree diagram rendering.
* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.
* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags.
* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

A.20. 18 to 19

* Updated the "keepalives" containers in part to address Michal Vasko's request to align with RFC 8071, and in part to better align to RFC 6520.
* Removed algorithm-mapping tables from the "TLS Common Model" section
* Removed the 'algorithm' node from the examples.
* Renamed both "client-certs" and "server-certs" to "ee-certs"
* Added a "Note to Reviewers" note to first page.

A.21. 19 to 20

* Modified the 'must' expression in the "ietf-tls-client:server-authention" node to cover the "raw-public-keys" and "psks" nodes also.

   * Added a "must 'ca-certs or ee-certs or raw-public-keys or psks'" statement to the ietf-tls-server:client-authentication" node.

   * Added "mandatory true" to "choice auth-type" and a "presence" statement to its ancestor.

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

   * Moved the "ietf-ssh-common" module section to proceed the other two module sections.

   * Updated the Security Considerations section.

A.22. 20 to 21

   * Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Mehmet Ersue, Balazs Kovacs, Radek Krejci, David Lamparter, Ladislav Lhotka, Alan Luchuk, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Michal Vasko, Bert Wijnen, and Liang Xia.

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Gary Wu
Cisco Systems

Email: garywu@cisco.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-41

Abstract

This document presents four YANG 1.1 modules. Three IETF modules, and one supporting IANA module.

The three IETF modules are: ietf-tls-common, ietf-tls-client, and ietf-tls-server. The "ietf-tls-client" and "ietf-tls-server" modules are the primary productions of this work, supporting the configuration and monitoring of TLS clients and servers.

The IANA module is: iana-tls-cipher-suite-algs. This module defines YANG enumerations providing support for an IANA-maintained algorithm registry.

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 draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* FFFF --> 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:

* 2024-03-16 --> the publication date of this draft

The "Relation to other RFCs" section Section 1.2 contains the text "one or more YANG modules" and, later, "modules". This text is sourced from a file in a context where it is unknown how many modules a draft defines. The text is not wrong as is, but it may be improved by stating more directly how many modules are defined.

The "Relation to other RFCs" section Section 1.2 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

The following Appendix sections are to be removed prior to publication:

* Appendix A.1. Initial Module for the "TLS Cipher Suites" Registry

* Appendix B. Change Log

Status of This Memo

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

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

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

This Internet-Draft will expire on 17 September 2024.

Table of Contents

1. Introduction

This document presents four YANG 1.1 [RFC7950] modules. Three "IETF" modules and one "IANA" module.

The three IETF modules are ietf-tls-common (Section 2), ietf-tls-client (Section 3), and ietf-tls-server (Section 4). The "ietf-tls-client" and "ietf-tls-server" modules are the primary productions of this work, supporting the configuration and monitoring of TLS clients and servers.

The groupings defined in this document are expected to be used in conjunction with the groupings defined in an underlying transport-level module, such as the groupings defined in [I-D.ietf-netconf-tcp-client-server]. The transport-level data model enables the configuration of transport-level values such as a remote address, a remote port, a local address, and a local port.

The IANA module is iana-tls-cipher-suite-algs (Appendix A.1). This module defines YANG enumerations providing support for an IANA-maintained algorithm registry.

This document assumes that the IANA module exists, and presents a script in Appendix A that IANA may use to generate the YANG module. This document does not publish initial version of this module. IANA publishes this module.

1.1. Regarding the IETF Modules

The three IETF modules define features and groupings to model "generic" TLS clients and TLS servers, where "generic" should be interpreted as "least common denominator" rather than "complete." Basic TLS protocol support is afforded by these modules, leaving configuration of advance features to augmentations made by consuming modules.

It is intended that the YANG groupings will be used by applications needing to configure TLS client and server protocol stacks. For instance, these groupings are used to help define the data model for HTTPS [RFC2818] and NETCONF over TLS [RFC7589] based clients and servers in [I-D.ietf-netconf-http-client-server] and [I-D.ietf-netconf-netconf-client-server] respectively.

The ietf-tls-client and ietf-tls-server YANG modules each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy for how the underlying TCP

connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "tls-server-grouping" grouping for the TLS parts it provides, while adding data nodes for the TCP-level call-home configuration.

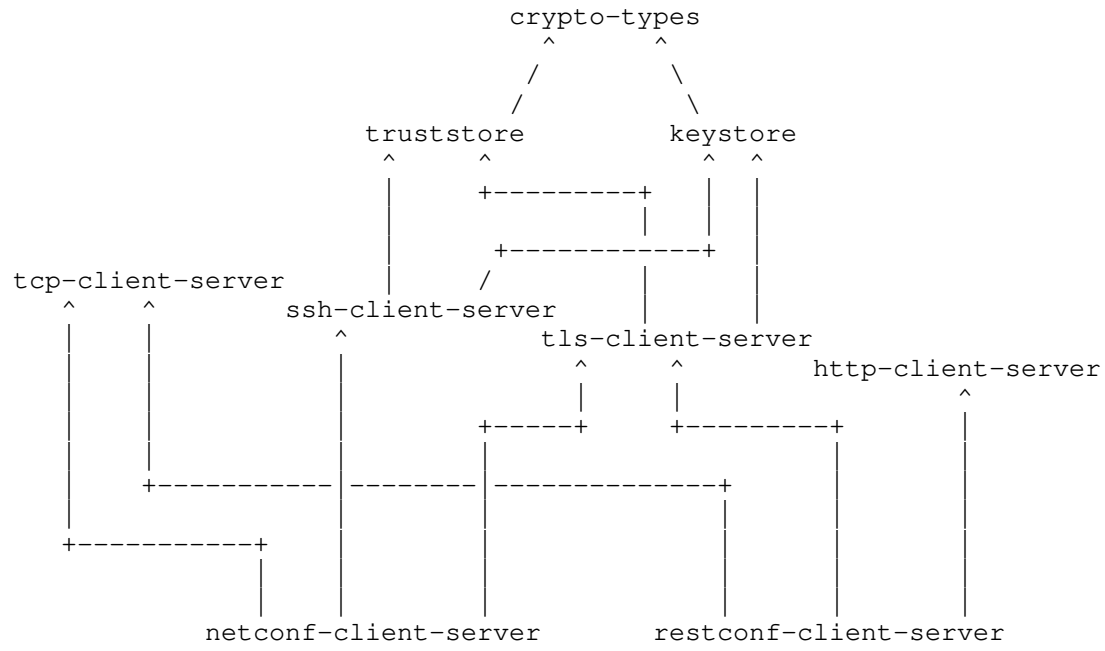
Both TLS 1.2 and TLS 1.3 may be configured. TLS 1.2 [RFC5246] is obsoleted by TLS 1.3 [RFC8446] but still in common use, and hence its "feature" statement is marked "status deprecated".

1.2. Relation to other RFCs

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

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



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

Table 1: Label in Diagram to RFC Mapping

1.3. 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.

1.4. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational> (Section 5.3 of [RFC8342]), and <system> [I-D.ietf-netmod-system-config], if implemented.

1.5. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-tls-common" Module

The TLS common model presented in this section contains features and groupings common to both TLS clients and TLS servers. The "hello-params-grouping" grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for TLS clients and TLS servers that are capable of doing so and may serve to make TLS clients and TLS servers compliant with local security policies. This model supports both TLS 1.2 [RFC5246] and TLS 1.3 [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the "hello-params-grouping" grouping should include three parameters for configuring its permitted TLS algorithms, which are: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups.

2.1. Data Model Overview

This section provides an overview of the "ietf-tls-common" module in terms of its features, identities, and groupings.

2.1.1. Features

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

```
Features:
  +-- tls12
  +-- tls13
  +-- hello-params
  +-- asymmetric-key-pair-generation
  +-- supported-algorithms
```

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

Please refer to the YANG module for a description of each feature.

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-tls-common" module:

```
Identities:
  +-- tls-version-base
     +-- tls12
     +-- tls13
```

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

Comments:

* The diagram shows that there are two base identities.
* One base identity is used to specific TLS versions, while the other is used to specify cipher-suites.
* These base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of things, rather than a specific thing.

2.1.3. Groupings

The "ietf-tls-common" module defines the following "grouping" statement:

* hello-params-grouping

This grouping is presented in the following subsection.

2.1.3.1. The "hello-params-grouping" Grouping

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

```
  grouping hello-params-grouping:
    +-- tls-versions
    |  +-- min?   identityref
    |  +-- max?   identityref
    +-- cipher-suites
       +-- cipher-suite*   tlscsa:tls-cipher-suite-algorithm
```

Comments:

* This grouping is used by both the "tls-client-grouping" and the "tls-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.

* This grouping enables client and server configurations to specify the TLS versions and cipher suites that are to be used when establishing TLS sessions.

* The "cipher-suites" list is "ordered-by user".

2.1.4. Protocol-accessible Nodes

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

```
module: ietf-tls-common
  +--ro supported-algorithms {algorithm-discovery}?
     +--ro supported-algorithm*   tlscsa:tls-cipher-suite-algorithm

  rpcs:
    +---x generate-asymmetric-key-pair
            {asymmetric-key-pair-generation}?
       +---w input
       |  +---w algorithm
       |  |       tlscsa:tls-cipher-suite-algorithm
       |  +---w num-bits?               uint16
       |  +---w private-key-encoding
       |     +---w (private-key-encoding)
       |        +--:(cleartext) {ct:cleartext-private-keys}?
       |        |  +---w cleartext?   empty
       |        +--:(encrypted) {ct:encrypted-private-keys}?
       |        |  +---w encrypted
       |        |     +---w ks:encrypted-by-grouping
       |        +--:(hidden) {ct:hidden-private-keys}?
       |           +---w hidden?      empty
       +--ro output
          +--ro (key-or-hidden)?
             +--:(key)
             |  +---u ct:asymmetric-key-pair-grouping
             +--:(hidden)
                +--ro location?
                        instance-identifier
```

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-tls-common" module are limited to the "supported-algorithms" container, which is constrained by the "algorithm-discovery" feature, and the RPC "generate-asymmetric-key-pair", which is constrained by the "asymmetric-key-pair-generation" feature.

* The "encrypted-by-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-keystore].

* The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.6 of [I-D.ietf-netconf-crypto-types].

2.2. Example Usage

The following example illustrates the "hello-params-grouping' grouping when populated with some data.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<hello-params
   xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common"
   xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
  <tls-versions>
    <min>tlscmn:tls12</min>
    <max>tlscmn:tls13</max>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</cipher-suite>
    <cipher-suite>TLS_DHE_RSA_WITH_AES_128_CBC_SHA256</cipher-suite>
    <cipher-suite>TLS_RSA_WITH_3DES_EDE_CBC_SHA</cipher-suite>
  </cipher-suites>
</hello-params>
```

The following example illustrates operational state data indicating the TLS algorithms supported by the server.

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

   <supported-algorithms
     xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common">
     <supported-algorithm>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</support\
   ed-algorithm>
     <supported-algorithm>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384</supp\
   orted-algorithm>
     <supported-algorithm>TLS_DHE_RSA_WITH_AES_128_CBC_SHA256</supporte\
   d-algorithm>
     <supported-algorithm>TLS_RSA_WITH_3DES_EDE_CBC_SHA</supported-algo\
   rithm>
     <supported-algorithm>TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384</suppor\
   ted-algorithm>
     <supported-algorithm>TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256</su\
   pported-algorithm>
     <supported-algorithm>TLS_ECCPWD_WITH_AES_256_GCM_SHA384</supported\
   -algorithm>
     <supported-algorithm>TLS_PSK_WITH_AES_256_CCM</supported-algorithm>
     <supported-algorithm>TLS_PSK_WITH_AES_256_CCM_8</supported-algorit\
   hm>
     <supported-algorithm>TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384</sup\
   ported-algorithm>
     <supported-algorithm>TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384</support\
   ed-algorithm>
     <supported-algorithm>TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA</supported\
   -algorithm>
     <supported-algorithm>TLS_DH_DSS_WITH_AES_128_GCM_SHA256</supported\
   -algorithm>
   </supported-algorithms>
```

The following example illustrates the "generate-asymmetric-key-pair" RPC.

REQUEST

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

   <rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <generate-asymmetric-key-pair
       xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common">
       <algorithm>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</algorithm>
       <num-bits>521</num-bits>
       <private-key-encoding>
         <encrypted>
           <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-key-re\
   f>
         </encrypted>
       </private-key-encoding>
     </generate-asymmetric-key-pair>
   </rpc>

   RESPONSE

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

   <rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
     xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
     <tlscmn:public-key-format>ct:subject-public-key-info-format</tlscm\
   n:public-key-format>
     <tlscmn:public-key>BASE64VALUE=</tlscmn:public-key>
     <tlscmn:private-key-format>ct:ec-private-key-format</tlscmn:privat\
   e-key-format>
     <tlscmn:cleartext-private-key>BASE64VALUE=</tlscmn:cleartext-priva\
   te-key>
   </rpc-reply>
```

2.3. YANG Module

This YANG module has a normative references to [RFC5288], [RFC5289], [RFC8422], and FIPS PUB 180-4.

This YANG module has a informative references to [RFC5246], and [RFC8446].

```
   <CODE BEGINS> file "ietf-tls-common@2024-03-16.yang"
```

```
   module ietf-tls-common {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
     prefix tlscmn;

     import iana-tls-cipher-suite-algs {
       prefix tlscsa;
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and SSH Servers";
     }

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

     import ietf-keystore {
       prefix ks;
       reference
         "RFC CCCC: A YANG Data Model for a Keystore";
     }

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

     contact
       "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        WG Web:   https://datatracker.ietf.org/wg/netconf
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Jeff Hartley <mailto:intensifysecurity@gmail.com>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

      description
       "This module defines a common features and groupings for
        Transport Layer Security (TLS).

        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 FFFF
```

```
        (https://www.rfc-editor.org/info/rfcFFFF); 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 2024-03-16 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Features

      feature tls12 {
        description
          "TLS Protocol Version 1.2 is supported. TLS 1.2 is obsolete
           and thus it is NOT RECOMMENDED to enable this feature.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2";
      }

      feature tls13 {
        description
          "TLS Protocol Version 1.3 is supported.";
        reference
          "RFC 8446: The Transport Layer Security (TLS)
                     Protocol Version 1.3";
      }

      feature hello-params {
        description
          "TLS hello message parameters are configurable.";
      }

      feature algorithm-discovery {
        description
          "Indicates that the server implements the
           'supported-algorithms' container.";
      }

      feature asymmetric-key-pair-generation {
```

```
      description
        "Indicates that the server implements the
         'generate-asymmetric-key-pair' RPC.";
    }

    // Identities

    identity tls-version-base {
      description
        "Base identity used to identify TLS protocol versions.";
    }

    identity tls12 {
      if-feature "tls12";
      base tls-version-base;
      description
        "TLS Protocol Version 1.2.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    identity tls13 {
      if-feature "tls13";
      base tls-version-base;
      description
        "TLS Protocol Version 1.3.";
      reference
        "RFC 8446: The Transport Layer Security (TLS)
                   Protocol Version 1.3";
    }

    // Typedefs

    typedef epsk-supported-hash {
      type enumeration {
        enum sha-256 {
          description
            "The SHA-256 Hash.";
        }
        enum sha-384 {
          description
            "The SHA-384 Hash.";
        }
      }
      description
        "As per Section 4.2.11 of RFC 8446, the hash algorithm
         supported by an instance of an External Pre-Shared
```

```
          Key (EPSK).";
      reference
        "RFC 8446: The Transport Layer Security (TLS)
                   Protocol Version 1.3";
    }


    // Groupings

    grouping hello-params-grouping {
      description
        "A reusable grouping for TLS hello message parameters.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2
         RFC 8446: The Transport Layer Security (TLS) Protocol
                   Version 1.3";
      container tls-versions {
        description
          "Parameters limiting which TLS versions, amongst
           those enabled by 'features', are presented during
           the TLS handshake.";
        leaf min {
          type identityref {
            base tls-version-base;
          }
          description
            "If not specified, then there is no configured
             minimum version.";
        }
        leaf max {
          type identityref {
            base tls-version-base;
          }
          description
            "If not specified, then there is no configured
             maximum version.";
        }
      }
      container cipher-suites {
        description
          "Parameters regarding cipher suites.";
        leaf-list cipher-suite {
          type tlscsa:tls-cipher-suite-algorithm;
          ordered-by user;
          description
            "Acceptable cipher suites in order of descending
             preference.  The configured host key algorithms should
```

```
            be compatible with the algorithm used by the configured
            private key.  Please see Section 5 of RFC FFFF for
            valid combinations.

            If this leaf-list is not configured (has zero elements)
            the acceptable cipher suites are implementation-
            defined.";
          reference
            "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
        }
      }
    } // hello-params-grouping


    // Protocol-accessible Nodes

    container supported-algorithms {
      if-feature "algorithm-discovery";
      config false;
      description
        "A container for a list of cipher suite algorithms supported
         by the server.";
      leaf-list supported-algorithm {
        type tlscsa:tls-cipher-suite-algorithm;
        description
          "A cipher suite algorithm supported by the server.";
      }
    }

    rpc generate-asymmetric-key-pair {
      if-feature "asymmetric-key-pair-generation";
      description
        "Requests the device to generate an asymmetric-key-pair
         key using the specified key algorithm.";
      input {
        leaf algorithm {
          type tlscsa:tls-cipher-suite-algorithm;
          mandatory true;
          description
            "The cipher suite algorithm that the generated key is
             to work with.  Implementations derive the public key
             algorithm from the cipher suite algorithm.  Example:
             cipher suite 'tls-rsa-with-aes-256-cbc-sha256' maps
             to the RSA public key.";
        }
        leaf num-bits {
          type uint16;
          description
```

```
            "Specifies the number of bits in the key to create.
             For RSA keys, the minimum size is 1024 bits and
             the default is 3072 bits. Generally, 3072 bits is
             considered sufficient. DSA keys must be exactly 1024
             bits as specified by FIPS 186-2.  For elliptical
             keys, the 'num-bits' value determines the key length
             of the curve (e.g., 256, 384 or 521), where valid
             values supported by the server are conveyed via an
             unspecified mechanism.  For some public algorithms,
             the keys have a fixed length and thus the 'num-bits'
             value is not specified.";
          }
          container private-key-encoding {
            description
              "Indicates how the private key is to be encoded.";
            choice private-key-encoding {
              mandatory true;
              description
                "A choice amongst optional private key handling.";
              case cleartext {
                if-feature "ct:cleartext-private-keys";
                leaf cleartext {
                  type empty;
                  description
                    "Indicates that the private key is to be returned
                     as a cleartext value.";
                }
              }
              case encrypted {
                if-feature "ct:encrypted-private-keys";
                container encrypted {
                  description
                     "Indicates that the key is to be encrypted using
                      the specified symmetric or asymmetric key.";
                  uses ks:encrypted-by-grouping;
                }
              }
              case hidden {
                if-feature "ct:hidden-private-keys";
                leaf hidden {
                  type empty;
                  description
                    "Indicates that the private key is to be hidden.

                     Unlike the 'cleartext' and 'encrypt' options, the
                     key returned is a placeholder for an internally
                     stored key.  See the 'Support for Built-in Keys'
                     section in RFC CCCC for information about hidden
```

```
                    keys.";
                }
              }
            }
          }
        }
        output {
          choice key-or-hidden {
            case key {
              uses ct:asymmetric-key-pair-grouping;
            }
            case hidden {
              leaf location {
                type instance-identifier;
                description
                  "The location to where a hidden key was created.";
              }
            }
            description
              "The output can be either a key (for cleartext and
               encrypted keys) or the location to where the key
               was created (for hidden keys).";
          }
        }
      } // end generate-asymmetric-key-pair

    }
```

<CODE ENDS>

3. The "ietf-tls-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-tls-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-tls-client" module in terms of its features and groupings.

3.1.1. Features

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

```
Features:
  +-- tls-client-keepalives
  +-- client-ident-x509-cert
  +-- client-ident-raw-public-key
  +-- client-ident-psk
  +-- server-auth-x509-cert
  +-- server-auth-raw-public-key
  +-- server-auth-psk
```

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

Please refer to the YANG module for a description of each feature.

3.1.2. Groupings

The "ietf-tls-client" module defines the following "grouping" statement:

* tls-client-grouping

This grouping is presented in the following subsection.

3.1.2.1. The "tls-client-grouping" Grouping

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

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

     grouping tls-client-grouping:
       +-- client-identity!
       |  +-- (auth-type)
       |     +--:(certificate) {client-ident-x509-cert}?
       |     |  +-- certificate
       |     |     +---u ks:inline-or-keystore-end-entity-cert-with-key\
   -grouping
       |     +--:(raw-public-key) {client-ident-raw-public-key}?
       |     |  +-- raw-private-key
       |     |     +---u ks:inline-or-keystore-asymmetric-key-grouping
       |     +--:(tls12-psk) {client-ident-tls12-psk}?
       |     |  +-- tls12-psk
       |     |     +---u ks:inline-or-keystore-symmetric-key-grouping
       |     |     +-- id?
       |     |              string
       |     +--:(tls13-epsk) {client-ident-tls13-epsk}?
       |        +-- tls13-epsk
       |           +---u ks:inline-or-keystore-symmetric-key-grouping
       |           +-- external-identity
       |           |       string
       |           +-- hash?
       |           |       tlscmn:epsk-supported-hash
       |           +-- context?
       |           |       string
       |           +-- target-protocol?
       |           |       uint16
       |           +-- target-kdf?
       |                   uint16
       +-- server-authentication
       |  +-- ca-certs! {server-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- ee-certs! {server-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- raw-public-keys! {server-auth-raw-public-key}?
       |  |  +---u ts:inline-or-truststore-public-keys-grouping
       |  +-- tls12-psks?        empty {server-auth-tls12-psk}?
       |  +-- tls13-epsks?       empty {server-auth-tls13-epsk}?
       +-- hello-params {tlscmn:hello-params}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives {tls-client-keepalives}?
          +-- peer-allowed-to-send?   empty
          +-- test-peer-aliveness!
             +-- max-wait?       uint16
             +-- max-attempts?   uint8
```

Comments:

* The "client-identity" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures identity credentials, each enabled by a "feature" statement defined in Section 3.1.1.

* The "server-authentication" node configures trust anchors for authenticating the TLS server, with each option enabled by a "feature" statement.

* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the TLS server. The aliveness-test occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "inline-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-trust-anchors].
  - The "inline-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-tls-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "tls-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

The following configuration example uses inline-definitions for the client identity and server authentication:

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

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

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

    <!-- how this client will authenticate itself to the server -->
    <client-identity>
        <certificate>
            <inline-definition>
                <private-key-format>ct:rsa-private-key-format</priva\
te-key-format>
                <cleartext-private-key>BASE64VALUE=</cleartext-priva\
te-key>
                <cert-data>BASE64VALUE=</cert-data>
            </inline-definition>
        </certificate>
    </client-identity>

    <!-- which certificates will this client trust -->
    <server-authentication>
        <ca-certs>
            <inline-definition>
                <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>
            </inline-definition>
```

```
          </ca-certs>
          <ee-certs>
              <inline-definition>
                  <certificate>
                      <name>My Application #1</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
                  <certificate>
                      <name>My Application #2</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
              </inline-definition>
          </ee-certs>
          <raw-public-keys>
              <inline-definition>
                  <public-key>
                      <name>corp-fw1</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
                  <public-key>
                      <name>corp-fw2</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
              </inline-definition>
          </raw-public-keys>
          <tls12-psks/>
          <tls13-epsks/>
      </server-authentication>

      <keepalives>
          <test-peer-aliveness>
              <max-wait>30</max-wait>
              <max-attempts>3</max-attempts>
          </test-peer-aliveness>
      </keepalives>

  </tls-client>
```

The following configuration example uses central-keystore-references for the client identity and central-truststore-references for server authentication: from the keystore:

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

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <tls-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client">

       <!-- how this client will authenticate itself to the server -->
       <client-identity>
           <certificate>
               <central-keystore-reference>
                   <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                   <certificate>ex-rsa-cert</certificate>
               </central-keystore-reference>
           </certificate>
       </client-identity>

       <!-- which certificates will this client trust -->
       <server-authentication>
           <ca-certs>
               <central-truststore-reference>trusted-server-ca-certs</c\
   entral-truststore-reference>
           </ca-certs>
           <ee-certs>
               <central-truststore-reference>trusted-server-ee-certs</c\
   entral-truststore-reference>
           </ee-certs>
           <raw-public-keys>
               <central-truststore-reference>Raw Public Keys for TLS Se\
   rvers</central-truststore-reference>
           </raw-public-keys>
           <tls12-psks/>
           <tls13-epsks/>
       </server-authentication>

       <keepalives>
           <test-peer-aliveness>
               <max-wait>30</max-wait>
               <max-attempts>3</max-attempts>
           </test-peer-aliveness>
       </keepalives>

   </tls-client>
```

3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], and Informative references to [RFC5246], [RFC8446], [RFC9258] and [RFC9257].

<CODE BEGINS> file "ietf-tls-client@2024-03-16.yang"

```
module ietf-tls-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix tlsc;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
```

```
       "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        WG Web:   https://datatracker.ietf.org/wg/netconf
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Jeff Hartley <mailto:intensifysecurity@gmail.com>";

     description
       "This module defines reusable groupings for TLS clients that
        can be used as a basis for specific TLS client instances.

        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 FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); 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 2024-03-16 {
       description
         "Initial version";
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
     }

     // Features

     feature tls-client-keepalives {
       description
         "Per socket TLS keepalive parameters are configurable for
          TLS clients on the server implementing this feature.";
     }

     feature client-ident-x509-cert {
       description
         "Indicates that the client supports identifying itself
```

```
          using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }

      feature client-ident-raw-public-key {
        description
          "Indicates that the client supports identifying itself
           using raw public keys.";
        reference
          "RFC 7250:
             Using Raw Public Keys in Transport Layer Security (TLS)
             and Datagram Transport Layer Security (DTLS)";
      }

      feature client-ident-tls12-psk {
        if-feature "tlscmn:tls12";
        description
          "Indicates that the client supports identifying itself
           using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }

      feature client-ident-tls13-epsk {
        if-feature "tlscmn:tls13";
        description
          "Indicates that the client supports identifying itself
           using TLS-1.3 External PSKs (pre-shared keys).";
        reference
          "RFC 8446:
             The Transport Layer Security (TLS) Protocol Version 1.3";
      }

      feature server-auth-x509-cert {
        description
          "Indicates that the client supports authenticating servers
           using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }
```

```
    feature server-auth-raw-public-key {
      description
        "Indicates that the client supports authenticating servers
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature server-auth-tls12-psk {
      description
        "Indicates that the client supports authenticating servers
         using PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature server-auth-tls13-epsk {
      description
        "Indicates that the client supports authenticating servers
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    // Groupings

    grouping tls-client-grouping {
      description
        "A reusable grouping for configuring a TLS client without
         any consideration for how an underlying TCP session is
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tls-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container client-identity {
        nacm:default-deny-write;
```

```
        presence
          "Indicates that a TLS-level client identity has been
           configured.  This statement is present so the mandatory
           descendant do not imply that this node must be configured.";
        description
          "Identity credentials the TLS client MAY present when
           establishing a connection to a TLS server.  If not
           configured, then client authentication is presumed to
           occur in a protocol layer above TLS.  When configured,
           and requested by the TLS server when establishing a
           TLS session, these credentials are passed in the
           Certificate message defined in Section 7.4.2 of
           RFC 5246 and Section 4.4.2 in RFC 8446.";
        reference
          "RFC 5246: The Transport Layer Security (TLS)
                     Protocol Version 1.2
           RFC 8446: The Transport Layer Security (TLS)
                     Protocol Version 1.3
           RFC CCCC: A YANG Data Model for a Keystore";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst authentication types, of which one must
             be enabled (via its associated 'feature') and selected.";
          case certificate {
            if-feature "client-ident-x509-cert";
            container certificate {
              description
                "Specifies the client identity using a certificate.";
              uses
                "ks:inline-or-keystore-end-entity-cert-with-key-"
                + "grouping" {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
                     + '(public-key-format, "ct:subject-public-key-'
                     + 'info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                     + "central-keystore-reference/asymmetric-key" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                     + 'derived-from-or-self(deref(.)/../ks:public-'
                     + 'key-format, "ct:subject-public-key-info-'
                     + 'format")';
                }
              }
            }
          }
          case raw-public-key {
```

```
            if-feature "client-ident-raw-public-key";
            container raw-private-key {
              description
                "Specifies the client identity using a raw
                 private key.";
              uses ks:inline-or-keystore-asymmetric-key-grouping {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
                     + '(public-key-format, "ct:subject-public-key-'
                     + 'info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                     + "central-keystore-reference" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                     + 'derived-from-or-self(deref(.)/../ks:public-'
                     + 'key-format, "ct:subject-public-key-info-'
                     + 'format")';
                }
              }
            }
          }
          case tls12-psk {
            if-feature "client-ident-tls12-psk";
            container tls12-psk {
              description
                "Specifies the client identity using a PSK (pre-shared
                 or pairwise-symmetric key).";
              uses ks:inline-or-keystore-symmetric-key-grouping;
              leaf id {
                type string;
                description
                  "The key 'psk_identity' value used in the TLS
                   'ClientKeyExchange' message.";
                reference
                  "RFC 4279: Pre-Shared Key Ciphersuites for
                             Transport Layer Security (TLS)";
              }
            }
          }
          case tls13-epsk {
            if-feature "client-ident-tls13-epsk";
            container tls13-epsk {
              description
                "An External Pre-Shared Key (EPSK) is established
                or provisioned out-of-band, i.e., not from a TLS
                connection.  An EPSK is a tuple of (Base Key,
                External Identity, Hash).  External PSKs MUST NOT
                be imported for (D)TLS 1.2 or prior versions.  When
```

```
            PSKs are provisioned out of band, the PSK identity
            and the KDF hash algorithm to be used with the PSK
            MUST also be provisioned.

            The structure of this container is designed to
            satisfy the requirements of RFC 8446 Section
            4.2.11, the recommendations from Section 6 in
            RFC 9257, and the EPSK input fields detailed in
            Section 5.1 in RFC 9258.  The base-key is based
            upon ks:inline-or-keystore-symmetric-key-grouping
            in order to provide users with flexible and
            secure storage options.";
          reference
            "RFC 8446: The Transport Layer Security (TLS)
                       Protocol Version 1.3
             RFC 9257: Guidance for External Pre-Shared Key
                       (PSK) Usage in TLS
             RFC 9258: Importing External Pre-Shared Keys
                       (PSKs) for TLS 1.3";
          uses ks:inline-or-keystore-symmetric-key-grouping;
          leaf external-identity {
            type string;
            mandatory true;
            description
              "As per Section 4.2.11 of RFC 8446, and Section 4.1
               of RFC 9257, a sequence of bytes used to identify
               an EPSK. A label for a pre-shared key established
               externally.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3
               RFC 9257: Guidance for External Pre-Shared Key
                         (PSK) Usage in TLS";
          }
          leaf hash {
            type tlscmn:epsk-supported-hash;
            default sha-256;
            description
              "As per Section 4.2.11 of RFC 8446, for externally
               established PSKs, the Hash algorithm MUST be set
               when the PSK is established or default to SHA-256
               if no such algorithm is defined.  The server MUST
               ensure that it selects a compatible PSK (if any)
               and cipher suite.  Each PSK MUST only be used with
               a single hash function.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3";
```

```
              }
              leaf context {
                type string;
                description
                  "Per Section 5.1 of RFC 9258, context MUST include
                   the context used to determine the EPSK, if
                   any exists. For example, context may include
                   information about peer roles or identities
                   to mitigate Selfie-style reflection attacks.
                   Since the EPSK is a key derived from an external
                   protocol or sequence of protocols, context MUST
                   include a channel binding for the deriving
                   protocols [RFC5056].  The details of this
                   binding are protocol specfic and out of scope
                   for this document.";
                reference
                  "RFC 9258: Importing External Pre-Shared Keys
                              (PSKs) for TLS 1.3";
              }
              leaf target-protocol {
                type uint16;
                description
                  "As per Section 3 of RFC 9258, the protocol
                   for which a PSK is imported for use.";
                reference
                  "RFC 9258: Importing External Pre-Shared Keys
                              (PSKs) for TLS 1.3";
              }
              leaf target-kdf {
                type uint16;
                description
                  "As per Section 3 of RFC 9258, the KDF for
                   which a PSK is imported for use.";
                reference
                  "RFC 9258: Importing External Pre-Shared Keys
                              (PSKs) for TLS 1.3";
              }
            }
          }
        }
      } // container client-identity

      container server-authentication {
        nacm:default-deny-write;
        must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
          or tls13-epsks';
        description
          "Specifies how the TLS client can authenticate TLS servers.
```

```
           Any combination of credentials is additive and unordered.

           Note that no configuration is required for PSK (pre-shared
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../client-
           identity' node.";
         container ca-certs {
           if-feature "server-auth-x509-cert";
           presence
             "Indicates that CA certificates have been configured.
              This statement is present so the mandatory descendant
              nodes do not imply that this node must be configured.";
           description
             "A set of certificate authority (CA) certificates used by
              the TLS client to authenticate TLS server certificates.
              A server certificate is authenticated if it has a valid
              chain of trust to a configured CA certificate.";
           reference
             "RFC BBBB: A YANG Data Model for a Truststore";
           uses ts:inline-or-truststore-certs-grouping;
         }
         container ee-certs {
           if-feature "server-auth-x509-cert";
           presence
             "Indicates that EE certificates have been configured.
              This statement is present so the mandatory descendant
              nodes do not imply that this node must be configured.";
           description
             "A set of server certificates (i.e., end entity
              certificates) used by the TLS client to authenticate
              certificates presented by TLS servers.  A server
              certificate is authenticated if it is an exact
              match to a configured server certificate.";
           reference
             "RFC BBBB: A YANG Data Model for a Truststore";
           uses ts:inline-or-truststore-certs-grouping;
         }
         container raw-public-keys {
           if-feature "server-auth-raw-public-key";
           presence
             "Indicates that raw public keys have been configured.
              This statement is present so the mandatory descendant
              nodes do not imply that this node must be configured.";
           description
             "A set of raw public keys used by the TLS client to
              authenticate raw public keys presented by the TLS
              server.  A raw public key is authenticated if it
              is an exact match to a configured raw public key.";
```

```
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-public-keys-grouping {
            refine "inline-or-truststore/inline/inline-definition/"
                 + "public-key" {
              must 'derived-from-or-self(public-key-format,'
                 + ' "ct:subject-public-key-info-format")';
            }
            refine "inline-or-truststore/central-truststore/"
                 + "central-truststore-reference" {
              must 'not(deref(.)/../ts:public-key/ts:public-key-'
                 + 'format[not(derived-from-or-self(., "ct:subject-'
                 + 'public-key-info-format"))])';
            }
          }
        }
        leaf tls12-psks {
          if-feature "server-auth-tls12-psk";
          type empty;
          description
            "Indicates that the TLS client can authenticate TLS servers
             using configured PSKs (pre-shared or pairwise-symmetric
             keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'client-identity'
             node.";
        }
        leaf tls13-epsks {
          if-feature "server-auth-tls13-epsk";
          type empty;
          description
            "Indicates that the TLS client can authenticate TLS servers
             using configured external PSKs (pre-shared keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'client-identity'
             node.";
        }
      } // container server-authentication

      container hello-params {
        nacm:default-deny-write;
        if-feature "tlscmn:hello-params";
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      } // container hello-params
```

```
      container keepalives {
        nacm:default-deny-write;
        if-feature "tls-client-keepalives";
        description
          "Configures the keepalive policy for the TLS client.";
        leaf peer-allowed-to-send {
          type empty;
          description
            "Indicates that the remote TLS server is allowed to send
             HeartbeatRequest messages, as defined by RFC 6520
             to this TLS client.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
        }
        container test-peer-aliveness {
          presence
            "Indicates that the TLS client proactively tests the
             aliveness of the remote TLS server.";
          description
            "Configures the keep-alive policy to proactively test
             the aliveness of the TLS server.  An unresponsive
             TLS server is dropped after approximately max-wait
             * max-attempts seconds.  The TLS client MUST send
             HeartbeatRequest messages, as defined by RFC 6520.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "30";
            description
              "Sets the amount of time in seconds after which if
               no data has been received from the TLS server, a
               TLS-level message will be sent to test the
               aliveness of the TLS server.";
          }
          leaf max-attempts {
            type uint8;
            default "3";
            description
              "Sets the maximum number of sequential keep-alive
               messages that can fail to obtain a response from
               the TLS server before assuming the TLS server is
               no longer alive.";
```

```
          }
        }
      }
    } // grouping tls-client-grouping

  }
```

<CODE ENDS>

4. The "ietf-tls-server" Module

This section defines a YANG 1.1 module called "ietf-tls-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-tls-server" module in terms of its features and groupings.

4.1.1. Features

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

```
Features:
  +-- tls-server-keepalives
  +-- server-ident-x509-cert
  +-- server-ident-raw-public-key
  +-- server-ident-psk
  +-- client-auth-supported
  +-- client-auth-x509-cert
  +-- client-auth-raw-public-key
  +-- client-auth-psk
```

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

Please refer to the YANG module for a description of each feature.

4.1.2. Groupings

The "ietf-tls-server" module defines the following "grouping" statement:

* tls-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "tls-server-grouping" Grouping

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

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

     grouping tls-server-grouping:
       +-- server-identity
       |  +-- (auth-type)
       |     +--:(certificate) {server-ident-x509-cert}?
       |     |  +-- certificate
       |     |     +---u ks:inline-or-keystore-end-entity-cert-with-key\
   -grouping
       |     +--:(raw-private-key) {server-ident-raw-public-key}?
       |     |  +-- raw-private-key
       |     |     +---u ks:inline-or-keystore-asymmetric-key-grouping
       |     +--:(tls12-psk) {server-ident-tls12-psk}?
       |     |  +-- tls12-psk
       |     |     +---u ks:inline-or-keystore-symmetric-key-grouping
       |     |     +-- id-hint?
       |     |           string
       |     +--:(tls13-epsk) {server-ident-tls13-epsk}?
       |        +-- tls13-epsk
       |           +---u ks:inline-or-keystore-symmetric-key-grouping
       |           +-- external-identity
       |           |       string
       |           +-- hash?
       |           |       tlscmn:epsk-supported-hash
       |           +-- context?
       |           |       string
       |           +-- target-protocol?
       |           |       uint16
       |           +-- target-kdf?
       |                   uint16
       +-- client-authentication! {client-auth-supported}?
       |  +-- ca-certs! {client-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- ee-certs! {client-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- raw-public-keys! {client-auth-raw-public-key}?
       |  |  +---u ts:inline-or-truststore-public-keys-grouping
       |  +-- tls12-psks?        empty {client-auth-tls12-psk}?
       |  +-- tls13-epsks?       empty {client-auth-tls13-epsk}?
       +-- hello-params {tlscmn:hello-params}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives {tls-server-keepalives}?
          +-- peer-allowed-to-send?   empty
          +-- test-peer-aliveness!
             +-- max-wait?       uint16
             +-- max-attempts?   uint8
```

Comments:

* The "server-identity" node configures identity credentials, each of which is enabled by a "feature".

* The "client-authentication" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures trust anchors for authenticating the TLS client, with each option enabled by a "feature" statement.

* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a flag enabling the TLS client to test the aliveness of the TLS server, as well as a "presence" container for testing the aliveness of the TLS client. The aliveness-tests occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "inline-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-trust-anchors].
  - The "inline-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

4.1.3. Protocol-accessible Nodes

The "ietf-tls-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

4.2. Example Usage

This section presents two examples showing the "tls-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

The following configuration example uses inline-definitions for the server identity and client authentication:

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

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

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

    <!-- how this server will authenticate itself to the client -->
    <server-identity>
        <certificate>
          <inline-definition>
              <private-key-format>ct:rsa-private-key-format</private\
-key-format>
              <cleartext-private-key>BASE64VALUE=</cleartext-private\
-key>
              <cert-data>BASE64VALUE=</cert-data>
            </inline-definition>
        </certificate>
    </server-identity>

    <!-- which certificates will this server trust -->
    <client-authentication>
        <ca-certs>
            <inline-definition>
                <certificate>
                    <name>Identity Cert Issuer #1</name>
                    <cert-data>BASE64VALUE=</cert-data>
                </certificate>
                <certificate>
                    <name>Identity Cert Issuer #2</name>
                    <cert-data>BASE64VALUE=</cert-data>
                </certificate>
            </inline-definition>
```

```
          </ca-certs>
          <ee-certs>
              <inline-definition>
                  <certificate>
                      <name>Application #1</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
                  <certificate>
                      <name>Application #2</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
              </inline-definition>
          </ee-certs>
          <raw-public-keys>
              <inline-definition>
                  <public-key>
                      <name>User A</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
                  <public-key>
                      <name>User B</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
              </inline-definition>
          </raw-public-keys>
          <tls12-psks/>
          <tls13-epsks/>
      </client-authentication>

      <keepalives>
          <peer-allowed-to-send/>
      </keepalives>

  </tls-server>
```

The following configuration example uses central-keystore-references for the server identity and central-truststore-references for client authentication: from the keystore:

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

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">

       <!-- how this server will authenticate itself to the client -->
       <server-identity>
           <certificate>
               <central-keystore-reference>
                   <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                   <certificate>ex-rsa-cert</certificate>
               </central-keystore-reference>
           </certificate>
       </server-identity>

       <!-- which certificates will this server trust -->
       <client-authentication>
           <ca-certs>
               <central-truststore-reference>trusted-client-ca-certs</c\
   entral-truststore-reference>
           </ca-certs>
           <ee-certs>
               <central-truststore-reference>trusted-client-ee-certs</c\
   entral-truststore-reference>
           </ee-certs>
           <raw-public-keys>
               <central-truststore-reference>Raw Public Keys for TLS Cl\
   ients</central-truststore-reference>
           </raw-public-keys>
           <tls12-psks/>
           <tls13-epsks/>
       </client-authentication>

       <keepalives>
           <peer-allowed-to-send/>
       </keepalives>

   </tls-server>
```

4.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], and Informative references to [RFC5246], [RFC8446], [RFC9258] and [RFC9257].

```
<CODE BEGINS> file "ietf-tls-server@2024-03-16.yang"

module ietf-tls-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix tlss;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

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

  description
    "This module defines reusable groupings for TLS servers that
```

```
        can be used as a basis for specific TLS server instances.

        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 FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); 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 2024-03-16 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Features

      feature tls-server-keepalives {
        description
          "Per socket TLS keepalive parameters are configurable for
           TLS servers on the server implementing this feature.";
      }

      feature server-ident-x509-cert {
        description
          "Indicates that the server supports identifying itself
           using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }
```

```
    feature server-ident-raw-public-key {
      description
        "Indicates that the server supports identifying itself
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature server-ident-tls12-psk {
      if-feature "tlscmn:tls12";
      description
        "Indicates that the server supports identifying itself
         using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature server-ident-tls13-epsk {
      if-feature "tlscmn:tls13";
      description
        "Indicates that the server supports identifying itself
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    feature client-auth-supported {
      description
        "Indicates that the configuration for how to authenticate
         clients can be configured herein.  TLS-level client
         authentication may not be needed when client authentication
         is expected to occur only at another protocol layer.";
    }

    feature client-auth-x509-cert {
      description
        "Indicates that the server supports authenticating clients
         using X.509 certificates.";
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile";
    }
```

```
    feature client-auth-raw-public-key {
      description
        "Indicates that the server supports authenticating clients
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature client-auth-tls12-psk {
      description
        "Indicates that the server supports authenticating clients
         using PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature client-auth-tls13-epsk {
      description
        "Indicates that the server supports authenticating clients
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    // Groupings

    grouping tls-server-grouping {
      description
        "A reusable grouping for configuring a TLS server without
         any consideration for how underlying TCP sessions are
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tls-server-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container server-identity {
        nacm:default-deny-write;
```

```
        description
          "A locally-defined or referenced end-entity certificate,
           including any configured intermediate certificates, the
           TLS server will present when establishing a TLS connection
           in its Certificate message, as defined in Section 7.4.2
           in RFC 5246 and Section 4.4.2 in RFC 8446.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2
           RFC 8446: The Transport Layer Security (TLS) Protocol
                     Version 1.3
           RFC CCCC: A YANG Data Model for a Keystore";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst authentication types, of which one must
             be enabled (via its associated 'feature') and selected.";
          case certificate {
            if-feature "server-ident-x509-cert";
            container certificate {
              description
                "Specifies the server identity using a certificate.";
              uses
                "ks:inline-or-keystore-end-entity-cert-with-key-"
                + "grouping" {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
                     + '(public-key-format,' + ' "ct:subject-public-'
                     + 'key-info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                     + "central-keystore-reference/asymmetric-key" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                     + 'derived-from-or-self(deref(.)/../ks:public-key'
                     + '-format, "ct:subject-public-key-info-format")';
                }
              }
            }
          }
          case raw-private-key {
            if-feature "server-ident-raw-public-key";
            container raw-private-key {
              description
                "Specifies the server identity using a raw
                 private key.";
              uses ks:inline-or-keystore-asymmetric-key-grouping {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
```

```
                    + '(public-key-format,' + ' "ct:subject-public-'
                    + 'key-info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                    + "central-keystore-reference" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                    + 'derived-from-or-self(deref(.)/../ks:public-key'
                    + '-format, "ct:subject-public-key-info-format")';
                }
              }
            }
          }
          case tls12-psk {
            if-feature "server-ident-tls12-psk";
            container tls12-psk {
              description
                "Specifies the server identity using a PSK (pre-shared
                 or pairwise-symmetric key).";
              uses ks:inline-or-keystore-symmetric-key-grouping;
              leaf id-hint {
                type string;
                description
                  "The key 'psk_identity_hint' value used in the TLS
                   'ServerKeyExchange' message.";
                reference
                  "RFC 4279: Pre-Shared Key Ciphersuites for
                             Transport Layer Security (TLS)";
              }
            }
          }
          case tls13-epsk {
            if-feature "server-ident-tls13-epsk";
            container tls13-epsk {
              description
                "An External Pre-Shared Key (EPSK) is established
                or provisioned out-of-band, i.e., not from a TLS
                connection.  An EPSK is a tuple of (Base Key,
                External Identity, Hash).  External PSKs MUST
                NOT be imported for (D)TLS 1.2 or prior versions.
                When PSKs are provisioned out of band, the PSK
                identity and the KDF hash algorithm to be used
                with the PSK MUST also be provisioned.

                The structure of this container is designed to
                satisfy the requirements of RFC 8446 Section
                4.2.11, the recommendations from Section 6 in
                RFC 9257, and the EPSK input fields detailed in
                Section 5.1 in RFC 9258.  The base-key is based
```

```
            upon ks:inline-or-keystore-symmetric-key-grouping
            in order to provide users with flexible and
            secure storage options.";
          reference
            "RFC 8446: The Transport Layer Security (TLS)
                       Protocol Version 1.3
             RFC 9257: Guidance for External Pre-Shared Key
                       (PSK) Usage in TLS
             RFC 9258: Importing External Pre-Shared Keys
                       (PSKs) for TLS 1.3";
          uses ks:inline-or-keystore-symmetric-key-grouping;
          leaf external-identity {
            type string;
            mandatory true;
            description
              "As per Section 4.2.11 of RFC 8446, and Section 4.1
               of RFC 9257, a sequence of bytes used to identify
               an EPSK. A label for a pre-shared key established
               externally.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3
               RFC 9257: Guidance for External Pre-Shared Key
                         (PSK) Usage in TLS";
          }
          leaf hash {
            type tlscmn:epsk-supported-hash;
            default sha-256;
            description
              "As per Section 4.2.11 of RFC 8446, for externally
               established PSKs, the Hash algorithm MUST be set
               when the PSK is established or default to SHA-256
               if no such algorithm is defined.  The server MUST
               ensure that it selects a compatible PSK (if any)
               and cipher suite.  Each PSK MUST only be used
               with a single hash function.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3";
          }
          leaf context {
            type string;
            description
              "Per Section 5.1 of RFC 9258, context MUST include
               the context used to determine the EPSK, if
               any exists. For example, context may include
               information about peer roles or identities
               to mitigate Selfie-style reflection attacks.
```

```
                    Since the EPSK is a key derived from an external
                    protocol or sequence of protocols, context MUST
                    include a channel binding for the deriving
                    protocols [RFC5056].  The details of this
                    binding are protocol specfic and out of scope
                    for this document.";
                  reference
                    "RFC 9258: Importing External Pre-Shared Keys
                               (PSKs) for TLS 1.3";
                }
                leaf target-protocol {
                  type uint16;
                  description
                    "As per Section 3.1 of RFC 9258, the protocol
                     for which a PSK is imported for use.";
                  reference
                    "RFC 9258: Importing External Pre-Shared Keys
                               (PSKs) for TLS 1.3";
                }
                leaf target-kdf {
                  type uint16;
                  description
                    "As per Section 3 of RFC 9258, the KDF for
                     which a PSK is imported for use.";
                  reference
                    "RFC 9258: Importing External Pre-Shared Keys
                               (PSKs) for TLS 1.3";
                }
              }
            }
          }
        } // container server-identity

        container client-authentication {
          if-feature "client-auth-supported";
          nacm:default-deny-write;
          must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
            or tls13-epsks';
          presence
            "Indicates that client authentication is supported (i.e.,
             that the server will request clients send certificates).
             If not configured, the TLS server SHOULD NOT request the
             TLS clients provide authentication credentials.";
          description
            "Specifies how the TLS server can authenticate TLS clients.
             Any combination of credentials is additive and unordered.

             Note that no configuration is required for PSK (pre-shared
```

```
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../server-
           identity' node.";
        container ca-certs {
          if-feature "client-auth-x509-cert";
          presence
            "Indicates that CA certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of certificate authority (CA) certificates used by
             the TLS server to authenticate TLS client certificates.
             A client certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "client-auth-x509-cert";
          presence
            "Indicates that EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of client certificates (i.e., end entity
             certificates) used by the TLS server to authenticate
             certificates presented by TLS clients. A client
             certificate is authenticated if it is an exact
             match to a configured client certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
        container raw-public-keys {
          if-feature "client-auth-raw-public-key";
          presence
            "Indicates that raw public keys have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of raw public keys used by the TLS server to
             authenticate raw public keys presented by the TLS
             client.  A raw public key is authenticated if it
             is an exact match to a configured raw public key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-public-keys-grouping {
```

```
            refine "inline-or-truststore/inline/inline-definition/"
                 + "public-key" {
              must 'derived-from-or-self(public-key-format,'
                 + ' "ct:subject-public-key-info-format")';
            }
            refine "inline-or-truststore/central-truststore/"
                 + "central-truststore-reference" {
              must 'not(deref(.)/../ts:public-key/ts:public-key-'
                 + 'format[not(derived-from-or-self(., "ct:subject-'
                 + 'public-key-info-format"))])';
            }
          }
        }
        leaf tls12-psks {
          if-feature "client-auth-tls12-psk";
          type empty;
          description
            "Indicates that the TLS server can authenticate TLS clients
             using configured PSKs (pre-shared or pairwise-symmetric
             keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'server-identity'
             node.";
        }
        leaf tls13-epsks {
          if-feature "client-auth-tls13-epsk";
          type empty;
          description
            "Indicates that the TLS 1.3 server can authenticate TLS
             clients using configured external PSKs (pre-shared keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'server-identity'
             node.";
        }
      } // container client-authentication

      container hello-params {
        nacm:default-deny-write;
        if-feature "tlscmn:hello-params";
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      } // container hello-params

      container keepalives {
        nacm:default-deny-write;
```

```
        if-feature "tls-server-keepalives";
        description
          "Configures the keepalive policy for the TLS server.";
        leaf peer-allowed-to-send {
          type empty;
          description
            "Indicates that the remote TLS client is allowed to send
             HeartbeatRequest messages, as defined by RFC 6520
             to this TLS server.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
        }
        container test-peer-aliveness {
          presence
            "Indicates that the TLS server proactively tests the
             aliveness of the remote TLS client.";
          description
            "Configures the keep-alive policy to proactively test
             the aliveness of the TLS client.  An unresponsive
             TLS client is dropped after approximately max-wait
             * max-attempts seconds.";
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "30";
            description
              "Sets the amount of time in seconds after which if
               no data has been received from the TLS client, a
               TLS-level message will be sent to test the
               aliveness of the TLS client.";
          }
          leaf max-attempts {
            type uint8;
            default "3";
            description
              "Sets the maximum number of sequential keep-alive
               messages that can fail to obtain a response from
               the TLS client before assuming the TLS client is
               no longer alive.";
          }
        }
      } // container keepalives
    } // grouping tls-server-grouping

  }
```

<CODE ENDS>

5. Security Considerations

The three IETF YANG modules in this document define groupings and will not be deployed as standalone modules. Their security implications may be context dependent based on their use in other modules. The designers of modules which import these grouping must conduct their own analysis of the security considerations.

5.1. Considerations for the "iana-tls-cipher-suite-algs" Module

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

The "iana-tls-cipher-suite-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

This YANG module defines YANG enumerations, for a public IANA-maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module. IANA MAY deprecate and/or obsolete enumerations over time as needed to address security issues found in the algorithms.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. Considerations for the "ietf-tls-common" YANG Module

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

The "ietf-tls-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

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

This module defines the RPC "generate-asymmetric-key-pair" that may, if the "ct:cleartext-private-keys" feature is enabled, and the client requests it, return the private clear in cleartext form. It is NOT RECOMMENDED for private keys to pass the server's security perimeter.

This module does not define any actions or notifications, and thus the security consideration for such is not provided here.

5.3. Considerations for the "ietf-tls-client" YANG Module

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

The "ietf-tls-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.4. Considerations for the "ietf-tls-server" YANG Module

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

The "ietf-tls-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

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

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.

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

Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers four URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

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

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

6.2. The "YANG Module Names" Registry

This document registers four YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         iana-tls-cipher-suite-algs
namespace:    urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs
prefix:       tlscsa
reference:    RFC FFFF

name:         ietf-tls-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-common
prefix:       tlscmn
reference:    RFC FFFF

name:         ietf-tls-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-client
prefix:       tlsc
reference:    RFC FFFF

name:         ietf-tls-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-server
prefix:       tlss
reference:    RFC FFFF
```

6.3. Considerations for the "iana-tls-cipher-suite-algs" Module

This section follows the template defined in Section 4.30.3.1 of [I-D.ietf-netmod-rfc8407bis].

This document presents a script (see Appendix A) for IANA to use to generate the IANA-maintained "iana-tls-cipher-suite-algs" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add the following note to the registry:

> New values must not be directly added to the "iana-tls-cipher-suite-algs" YANG module. They must instead be added to the "TLS Cipher Suites" sub-registry of the "Transport Layer Security (TLS) Parameters" registry [IANA-CIPHER-ALGS].

When a value is added to the "TLS Cipher Suites" sub-registry, a new "enum" statement must be added to the "iana-tls-cipher-suite-algs" YANG module. The "enum" statement, and sub-statements thereof, should be defined as follows:

enum
: Replicates a name from the registry.

value
: Contains the decimal value of the IANA-assigned value.

status
: Include only if a registration has been deprecated or obsoleted. An IANA "Recommended" maps to YANG status "deprecated". Since the registry is unable to express a logical "MUST NOT" recommendation, there is no mapping to YANG status "obsolete", which is unfortunate given Moving single-DES and IDEA TLS ciphersuites to Historic (https://datatracker.ietf.org/doc/status-change-tls-des-idea-ciphers-to-historic) .

description
: Contains "Enumeration for the 'TLS_FOO' algorithm.", where "TLS_FOO" is a placeholder for the algorithm's name (e.g., "TLS_PSK_WITH_AES_256_CBC_SHA").

reference
: Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-tls-cipher-suite-algs" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements. The "revision" must have a "description" statement explaining why the the update occurred, and must have a "reference" substatement that points to the document defining the registry update that resulted in this change. For instance:

```
revision 2024-02-02 {
    description
        "This update reflect the update made to the underlying
         Foo Bar registry per RFC XXXX.";
    reference
        "RFC XXXX: Extend the Foo Bars Registry
                   to Support Something Important";
}
```

IANA is requested to add the following note to the "TLS Cipher Suites" sub-registry of the "Transport Layer Security (TLS) Parameters" registry [IANA-CIPHER-ALGS].

> When this registry is modified, the YANG module "iana-tls-cipher-suite-algs" [IANA-YANG-PARAMETERS] must be updated as defined in RFC FFFF.

An initial version of this module can be found in Appendix A.1.

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[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>.

[RFC2712] Medvinsky, A. and M. Hur, "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)", RFC 2712, DOI 10.17487/RFC2712, October 1999, <https://www.rfc-editor.org/info/rfc2712>.

[RFC4162] Lee, H.J., Yoon, J.H., and J.I. Lee, "Addition of SEED Cipher Suites to Transport Layer Security (TLS)", RFC 4162, DOI 10.17487/RFC4162, August 2005, <https://www.rfc-editor.org/info/rfc4162>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

[RFC4785] Blumenthal, U. and P. Goel, "Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)", RFC 4785, DOI 10.17487/RFC4785, January 2007, <https://www.rfc-editor.org/info/rfc4785>.

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, DOI 10.17487/RFC5054, November 2007, <https://www.rfc-editor.org/info/rfc5054>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <https://www.rfc-editor.org/info/rfc5289>.

[RFC5469] Eronen, P., Ed., "DES and IDEA Cipher Suites for Transport Layer Security (TLS)", RFC 5469, DOI 10.17487/RFC5469, February 2009, <https://www.rfc-editor.org/info/rfc5469>.

[RFC5487] Badra, M., "Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode", RFC 5487, DOI 10.17487/RFC5487, March 2009, <https://www.rfc-editor.org/info/rfc5487>.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <https://www.rfc-editor.org/info/rfc5489>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <https://www.rfc-editor.org/info/rfc5746>.

[RFC5932] Kato, A., Kanda, M., and S. Kanno, "Camellia Cipher Suites for TLS", RFC 5932, DOI 10.17487/RFC5932, June 2010, <https://www.rfc-editor.org/info/rfc5932>.

[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>.

[RFC6209] Kim, W., Lee, J., Park, J., and D. Kwon, "Addition of the ARIA Cipher Suites to Transport Layer Security (TLS)", RFC 6209, DOI 10.17487/RFC6209, April 2011, <https://www.rfc-editor.org/info/rfc6209>.

[RFC6367] Kanno, S. and M. Kanda, "Addition of the Camellia Cipher Suites to Transport Layer Security (TLS)", RFC 6367, DOI 10.17487/RFC6367, September 2011, <https://www.rfc-editor.org/info/rfc6367>.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <https://www.rfc-editor.org/info/rfc6655>.

[RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", RFC 7251, DOI 10.17487/RFC7251, June 2014, <https://www.rfc-editor.org/info/rfc7251>.

[RFC7507] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", RFC 7507, DOI 10.17487/RFC7507, April 2015, <https://www.rfc-editor.org/info/rfc7507>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[RFC7905] Langley, A., Chang, W., Mavrogiannopoulos, N., Strombergson, J., and S. Josefsson, "ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)", RFC 7905, DOI 10.17487/RFC7905, June 2016, <https://www.rfc-editor.org/info/rfc7905>.

[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>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[RFC8442] Mattsson, J. and D. Migault, "ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2", RFC 8442, DOI 10.17487/RFC8442, September 2018, <https://www.rfc-editor.org/info/rfc8442>.

[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>.

[RFC8492] Harkins, D., Ed., "Secure Password Ciphersuites for Transport Layer Security (TLS)", RFC 8492, DOI 10.17487/RFC8492, February 2019, <https://www.rfc-editor.org/info/rfc8492>.

[RFC8998] Yang, P., "ShangMi (SM) Cipher Suites for TLS 1.3", RFC 8998, DOI 10.17487/RFC8998, March 2021, <https://www.rfc-editor.org/info/rfc8998>.

[RFC9150] Cam-Winget, N. and J. Visoky, "TLS 1.3 Authentication and Integrity-Only Cipher Suites", RFC 9150, DOI 10.17487/RFC9150, April 2022, <https://www.rfc-editor.org/info/rfc9150>.

[RFC9189] Smyshlyaev, S., Ed., Belyavsky, D., and E. Alekseev, "GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.2", RFC 9189, DOI 10.17487/RFC9189, March 2022, <https://www.rfc-editor.org/info/rfc9189>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[I-D.ietf-netmod-rfc8407bis]
Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-09, 28 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-09>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-05, 21 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

[IANA-CIPHER-ALGS]
(IANA), I. A. N. A., "IANA "TLS Cipher Suites" Sub-registry of the "Transport Layer Security (TLS) Parameters" Registry", <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4>.

[IANA-YANG-PARAMETERS]
"YANG Parameters", n.d., <https://www.iana.org/assignments/yang-parameters>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

[RFC9257] Housley, R., Hoyland, J., Sethi, M., and C. A. Wood, "Guidance for External Pre-Shared Key (PSK) Usage in TLS", RFC 9257, DOI 10.17487/RFC9257, July 2022, <https://www.rfc-editor.org/info/rfc9257>.

[RFC9258] Benjamin, D. and C. A. Wood, "Importing External Pre-Shared Keys (PSKs) for TLS 1.3", RFC 9258, DOI 10.17487/RFC9258, July 2022, <https://www.rfc-editor.org/info/rfc9258>.

Appendix A. Script to Generate IANA-Maintained YANG Modules

This section is not Normative.

The Python https://www.python.org script contained in this section will create the IANA-maintained module described in this document.

Run the script using the command 'python gen-yang-modules.py', to produce the YANG module file in the current directory.

Be aware that the script does not attempt to copy the "revision" statements from the previous/current YANG module. Copying the revision statements must be done manually.

```
<CODE BEGINS>
=============== NOTE: '\\' line wrapping per RFC 8792 ===============

import re
import csv
import requests
import textwrap
import requests_cache
from io import StringIO
from datetime import datetime

# Metadata for the one YANG module produced by this script
MODULES = [
    {
```

```
        "csv_url": "https://www.iana.org/assignments/tls-parameters/\
  \tls-parameters-4.csv",
        "spaced_name": "cipher-suite",
        "hypenated_name": "cipher-suite",
        "prefix": "tlscsa",
    }
  ]


  def create_module_begin(module, f):

    # Define template for all four modules
    PREAMBLE_TEMPLATE="""
  module iana-tls-HNAME-algs {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-tls-HNAME-algs";
    prefix PREFIX;

    organization
      "Internet Assigned Numbers Authority (IANA)";

    contact
      "Postal: ICANN
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA  90094-2536
               United States of America
       Tel:    +1 310 301 5800
       Email:  iana@iana.org";

    description
      "This module defines enumerations for the Cipher Suite
       algorithms defined in the 'TLS Cipher Suites' sub-registry
       of the 'Transport Layer Security (TLS) Parameters' registry
       maintained by IANA.

       Copyright (c) YEAR 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).

       The initial version of this YANG module is part of RFC FFFF
       (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
       itself for full legal notices.
```

```
        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

    revision DATE {
      description
        "This initial version of the module was created using
         the script defined in RFC FFFF to reflect the contents
         of the SNAME algorithms registry maintained by IANA.";
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    typedef tls-HNAME-algorithm {
      type enumeration {
"""
    # Replacements
    rep = {
      "DATE": datetime.today().strftime('%Y-%m-%d'),
      "YEAR": datetime.today().strftime('%Y'),
      "SNAME": module["spaced_name"],
      "HNAME": module["hypenated_name"],
      "PREFIX": module["prefix"]
    }

    # Do the replacement
    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))
    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], PREAMBL\
\E_TEMPLATE)

    # Write preamble into the file
    f.write(text)


def create_module_body(module, f):

    # Fetch the current CSV file from IANA
    r = requests.get(module["csv_url"])
    assert r.status_code == 200, "Could not get " + module["csv_url"]

    # Parse each CSV line
    with StringIO(r.text) as csv_file:
        csv_reader = csv.DictReader(csv_file)
        for row in csv_reader:

            # Skip reserved algs
            if row["Description"].startswith("Unassigned"):
                continue
```

```
            # Skip unassigned algs
            if row["Description"].startswith("Reserved"):
                continue

            # Ensure this is the TLS line
            assert row["Description"].startswith("TLS_"), "Unrecogni\
zed description: '" + row["Description"] + "'"

            # Set the 'refs' and 'titles' lists
            if row["Reference"] == "":
                pass # skip when the Reference field is empty

            else:

                # There may be more than one ref
                refs = row["Reference"][1:-1]  # remove the '[' and \
\']' chars
                refs = refs.split("][")
                titles = []
                for ref in refs:

                    # Ascertain the ref's title
                    if ref.startswith("RFC"):

                        # Fetch the current BIBTEX entry
                        bibtex_url="https://datatracker.ietf.org/doc\
\/"+ ref.lower() + "/bibtex/"
                        r = requests.get(bibtex_url)
                        assert r.status_code == 200, "Could not GET \
\" + bibtex_url

                        # Append to 'titles' value from the "title" \
\line
                        for item in r.text.split("\n"):
                            if "title =" in item:
                                title = re.sub('.*{{(.*)}}.*', r'\g<\
\1>', item)
                                if title.startswith("ECDHE\_PSK"):
                                    title = re.sub("ECDHE\\\\_PSK", \
\"ECDHE_PSK", title)
                                titles.append(re.sub('.*{{(.*)}}.*',\
\ r'\g<1>', title))
                                break
                        else:
                            raise Exception("RFC title not found")


                        # Insert a space: "RFCXXXX" --> "RFC XXXX"
```

```
                        index = refs.index(ref)
                        refs[index] = "RFC " + ref[3:]

                    elif ref == "IESG Action 2018-08-16":

                        # Rewrite the ref value
                        index = refs.index(ref)
                        refs[index] = "IESG Action"

                        # Let title be something descriptive
                        titles.append("IESG Action 2018-08-16")

                    elif ref == "draft-irtf-cfrg-aegis-aead-08":

                        # Manually set the draft's title
                        titles.append("The AEGIS Family of Authentic\
   \ated Encryption Algorithms")

                    elif ref:
                        raise Exception(f'ref "{ref}" not found')

                    else:
                        raise Exception(f'ref missing: {row}')

            # Write out the enum
            f.write(f'      enum {row["Description"]} {{\n');
            if row["Recommended"] == 'N':
                f.write(f'        status deprecated;\n')
            f.write(f'        description\n')
            description = f'          "Enumeration for the \'{row["D\
   \escription"]}\' algorithm.";'
            description = textwrap.fill(description, width=69, subse\
   \quent_indent="           ")
            f.write(f'{description}\n')
            f.write('        reference\n')
            f.write('          "')
            if row["Reference"] == "":
                f.write('Missing in IANA registry.')
            else:
                ref_len = len(refs)
                for i in range(ref_len):
                    ref = refs[i]
                    f.write(f'{ref}:\n')
                    title = "             " + titles[i]
                    if i == ref_len - 1:
                        title += '";'
                    title = textwrap.fill(title, width=69, subsequen\
   \t_indent="             ")
```

```
                    f.write(f'{title}')
                    if i != ref_len - 1:
                        f.write('\n           ')
            f.write('\n')
            f.write('      }\n')


   def create_module_end(module, f):

       # Close out the enumeration, typedef, and module
       f.write("    }\n")
       f.write("    description\n")
       f.write(f'      "An enumeration for TLS {module["spaced_name"]} \
   \algorithms.";\n')
       f.write("  }\n")
       f.write('\n')
       f.write('}\n')


   def create_module(module):

       # Install cache for 8x speedup
       requests_cache.install_cache()

       # Ascertain yang module's name
       yang_module_name = "iana-tls-" + module["hypenated_name"] + "-al\
   \gs.yang"

       # Create yang module file
       with open(yang_module_name, "w") as f:
           create_module_begin(module, f)
           create_module_body(module, f)
           create_module_end(module, f)


   def main():
       for module in MODULES:
           create_module(module)


   if __name__ == "__main__":
       main()
   <CODE ENDS>
```

A.1. Initial Module for the "TLS Cipher Suites" Registry

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2024-03-16" reflects the day on which the extraction occurred. Applications SHOULD use the IANA-maintained module, not the module defined in this draft.

This YANG module has normative references to [RFC2712], [RFC4162], [RFC4279], [RFC4346], [RFC4785], [RFC5054], [RFC5246], [RFC5288], [RFC5289], [RFC5469], [RFC5487], [RFC5489], [RFC5746], [RFC5932], [RFC6209], [RFC6367], [RFC6655], [RFC7251], [RFC7507], [RFC7905], [RFC8422], [RFC8442], [RFC8446], [RFC8492], [RFC8998], [RFC9150], [RFC9189], and [RFC8340].

<CODE BEGINS> file "iana-tls-cipher-suite-algs@2024-03-16.yang"

```
module iana-tls-cipher-suite-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs";
  prefix tlscsa;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
     Tel:    +1 310 301 5800
     Email:  iana@iana.org";

  description
    "This module defines enumerations for the Cipher Suite
     algorithms defined in the 'TLS Cipher Suites' sub-registry
     of the 'Transport Layer Security (TLS) Parameters' registry
     maintained by IANA.

     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).
```

```
        The initial version of this YANG module is part of RFC FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
        itself for full legal notices.

        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

      revision 2024-03-16 {
        description
          "This initial version of the module was created using
           the script defined in RFC FFFF to reflect the contents
           of the cipher-suite algorithms registry maintained by IANA.";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      typedef tls-cipher-suite-algorithm {
        type enumeration {
          enum TLS_NULL_WITH_NULL_NULL {
            status deprecated;
            description
              "Enumeration for the 'TLS_NULL_WITH_NULL_NULL' algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_NULL_MD5 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_NULL_MD5' algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_NULL_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_NULL_SHA' algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_EXPORT_WITH_RC4_40_MD5 {
            status deprecated;
            description
```

```
            "Enumeration for the 'TLS_RSA_EXPORT_WITH_RC4_40_MD5'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version 1.1
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_RSA_WITH_RC4_128_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_RC4_128_MD5'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version 1.2
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_RSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version 1.2
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_RSA_WITH_IDEA_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_IDEA_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
```

```
        }
        enum TLS_RSA_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_EXPORT_WITH_DES40_CBC_SHA'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_RSA_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DH_DSS_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
```

```
        }
        enum TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DH_RSA_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA' algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
```

```
            1.1";
        }
        enum TLS_DHE_DSS_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA' algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DHE_RSA_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
```

```
            1.2";
        }
        enum TLS_DH_anon_EXPORT_WITH_RC4_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_EXPORT_WITH_RC4_40_MD5'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version 1.1
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_DH_anon_WITH_RC4_128_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_RC4_128_MD5'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version 1.2
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA' algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DH_anon_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DH_anon_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
```

```
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_KRB5_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_KRB5_WITH_IDEA_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_IDEA_CBC_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_DES_CBC_MD5 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_DES_CBC_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_3DES_EDE_CBC_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_3DES_EDE_CBC_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_RC4_128_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_RC4_128_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_KRB5_WITH_IDEA_CBC_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_IDEA_CBC_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA'
             algorithm.";
          reference
            "RFC 2712:
```

```
            Addition of Kerberos Cipher Suites to Transport Layer
            Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC4_40_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC4_40_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC2_CBC_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC2_CBC_40_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC4_40_MD5 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC4_40_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_NULL_SHA' algorithm.";
          reference
            "RFC 4785:
               Pre-Shared Key (PSK) Ciphersuites with NULL Encryption
               for Transport Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 4785:
               Pre-Shared Key (PSK) Ciphersuites with NULL Encryption
               for Transport Layer Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 4785:
               Pre-Shared Key (PSK) Ciphersuites with NULL Encryption
               for Transport Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
```

```
          enum TLS_DH_DSS_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_DSS_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_RSA_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_DSS_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_RSA_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_anon_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
```

```
            1.2";
        }
        enum TLS_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_DSS_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_DSS_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
```

```
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
          }
          enum TLS_DH_anon_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_256_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_NULL_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_NULL_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_AES_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_DSS_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_DSS_WITH_AES_128_CBC_SHA256'
```

```
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_DH_RSA_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_DSS_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_RSA_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_128_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA'
```

```
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_DH_DSS_WITH_AES_256_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_DSS_WITH_AES_256_CBC_SHA256'
           algorithm.";
        reference
```

```
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
          }
          enum TLS_DH_RSA_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_anon_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_anon_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_256_CBC_SHA256'
```

```
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_RSA_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA' algorithm.";
        reference
```

```
            "RFC 5932:
               Camellia Cipher Suites for TLS";
          }
          enum TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA' algorithm.";
            reference
              "RFC 5932:
                 Camellia Cipher Suites for TLS";
          }
          enum TLS_PSK_WITH_RC4_128_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_RC4_128_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)
               RFC 6347:
                 Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_PSK_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_PSK_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_PSK_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_AES_256_CBC_SHA'
```

```
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
      enum TLS_DHE_PSK_WITH_RC4_128_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_RC4_128_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)
           RFC 6347:
             Datagram Transport Layer Security Version 1.2";
      }
      enum TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
      enum TLS_DHE_PSK_WITH_AES_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_CBC_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
      enum TLS_DHE_PSK_WITH_AES_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
```

```
          enum TLS_RSA_PSK_WITH_RC4_128_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_RC4_128_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)
               RFC 6347:
                 Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_RSA_PSK_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_RSA_PSK_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_256_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_RSA_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
```

```
            "RFC 4162:
               Addition of SEED Cipher Suites to Transport Layer
               Security (TLS)";
          }
          enum TLS_DH_DSS_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_DSS_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_RSA_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_anon_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_SEED_CBC_SHA'
```

```
           algorithm.";
        reference
          "RFC 4162:
             Addition of SEED Cipher Suites to Transport Layer
             Security (TLS)";
      }
      enum TLS_RSA_WITH_AES_128_GCM_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_RSA_WITH_AES_256_GCM_SHA384 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_AES_256_GCM_SHA384'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 {
        description
          "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 {
        description
          "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_GCM_SHA384'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_DH_RSA_WITH_AES_128_GCM_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
```

```
        }
        enum TLS_DH_RSA_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_anon_WITH_AES_128_GCM_SHA256 {
          status deprecated;
```

```
          description
            "Enumeration for the 'TLS_DH_anon_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_anon_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_PSK_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_PSK_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_GCM_SHA384'
```

```
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_RSA_PSK_WITH_AES_128_GCM_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_PSK_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_RSA_PSK_WITH_AES_256_GCM_SHA384 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_PSK_WITH_AES_256_GCM_SHA384'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_PSK_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_PSK_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_PSK_WITH_AES_256_CBC_SHA384 {
        status deprecated;
        description
          "Enumeration for the 'TLS_PSK_WITH_AES_256_CBC_SHA384'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_PSK_WITH_NULL_SHA256 {
        status deprecated;
```

```
          description
            "Enumeration for the 'TLS_PSK_WITH_NULL_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_PSK_WITH_NULL_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_NULL_SHA384'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_NULL_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_NULL_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
```

```
          enum TLS_DHE_PSK_WITH_NULL_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_PSK_WITH_NULL_SHA384'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_AES_256_CBC_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_256_CBC_SHA384'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_NULL_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_NULL_SHA256'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_NULL_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_NULL_SHA384'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
```

```
            and AES Galois Counter Mode";
        }
        enum TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA256 {
```

```
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
```

```
          'TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
        reference
          "RFC 5932:
            Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
        reference
          "RFC 5932:
            Camellia Cipher Suites for TLS";
      }
      enum TLS_SM4_GCM_SM3 {
        status deprecated;
        description
          "Enumeration for the 'TLS_SM4_GCM_SM3' algorithm.";
        reference
          "RFC 8998:
            ShangMi (SM) Cipher Suites for TLS 1.3";
      }
      enum TLS_SM4_CCM_SM3 {
        status deprecated;
        description
          "Enumeration for the 'TLS_SM4_CCM_SM3' algorithm.";
        reference
          "RFC 8998:
            ShangMi (SM) Cipher Suites for TLS 1.3";
      }
      enum TLS_EMPTY_RENEGOTIATION_INFO_SCSV {
        status deprecated;
        description
          "Enumeration for the 'TLS_EMPTY_RENEGOTIATION_INFO_SCSV'
           algorithm.";
        reference
          "RFC 5746:
            Transport Layer Security (TLS) Renegotiation Indication
            Extension";
      }
      enum TLS_AES_128_GCM_SHA256 {
        description
          "Enumeration for the 'TLS_AES_128_GCM_SHA256' algorithm.";
        reference
          "RFC 8446:
            The Transport Layer Security (TLS) Protocol Version
            1.3";
      }
```

```
        enum TLS_AES_256_GCM_SHA384 {
          description
            "Enumeration for the 'TLS_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version
               1.3";
        }
        enum TLS_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the 'TLS_CHACHA20_POLY1305_SHA256'
             algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version
               1.3";
        }
        enum TLS_AES_128_CCM_SHA256 {
          description
            "Enumeration for the 'TLS_AES_128_CCM_SHA256' algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version
               1.3";
        }
        enum TLS_AES_128_CCM_8_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_AES_128_CCM_8_SHA256'
             algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version 1.3
             IESG Action:
               IESG Action 2018-08-16";
        }
        enum TLS_AEGIS_256_SHA512 {
          status deprecated;
          description
            "Enumeration for the 'TLS_AEGIS_256_SHA512' algorithm.";
          reference
            "draft-irtf-cfrg-aegis-aead-08:
               The AEGIS Family of Authenticated Encryption
               Algorithms";
        }
        enum TLS_AEGIS_128L_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_AEGIS_128L_SHA256' algorithm.";
          reference
            "draft-irtf-cfrg-aegis-aead-08:
               The AEGIS Family of Authenticated Encryption
               Algorithms";
        }
        enum TLS_FALLBACK_SCSV {
          status deprecated;
          description
            "Enumeration for the 'TLS_FALLBACK_SCSV' algorithm.";
          reference
            "RFC 7507:
               TLS Fallback Signaling Cipher Suite Value (SCSV) for
               Preventing Protocol Downgrade Attacks";
        }
        enum TLS_ECDH_ECDSA_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_ECDSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
```

```
        }
        enum TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
```

```
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA' algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_RSA_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_RSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
```

```
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_RSA_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_RSA_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_256_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDHE_RSA_WITH_NULL_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDHE_RSA_WITH_NULL_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
```

```
            Earlier";
        }
        enum TLS_ECDHE_RSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_anon_WITH_NULL_SHA {
```

```
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_NULL_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_anon_WITH_RC4_128_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_RC4_128_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and Earlier
               RFC 6347:
                 Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_anon_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_anon_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_AES_256_CBC_SHA'
```

```
            algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA' algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA' algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
```

```
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5289:
```

```
            TLS Elliptic Curve Cipher Suites with SHA-256/384 and
            AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 5289:
```

```
            TLS Elliptic Curve Cipher Suites with SHA-256/384 and
            AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5289:
```

```
            TLS Elliptic Curve Cipher Suites with SHA-256/384 and
            AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_PSK_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_NULL_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_NULL_SHA256'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_NULL_SHA384 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_NULL_SHA384'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_DSS_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_DSS_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
```

```
        }
        enum TLS_DH_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_DSS_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_DSS_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
```

```
            Addition of the ARIA Cipher Suites to Transport Layer
            Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
```

```
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384 {
```

```
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_ARIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_128_GCM_SHA256'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_ARIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_256_GCM_SHA384'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
```

```
        }
        enum TLS_DH_DSS_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_DSS_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
```

```
            Addition of the ARIA Cipher Suites to Transport Layer
            Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
```

```
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_256_GCM_SHA384 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
```

```
        }
        enum TLS_ECDHE_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
```

```
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
```

```
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
```

```
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DH_DSS_WITH_CAMELLIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_DSS_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
```

```
        }
        enum TLS_DH_DSS_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_DSS_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DH_anon_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DH_anon_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384'
             algorithm.";
```

```
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_PSK_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_PSK_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_256_GCM_SHA384 {
```

```
            status deprecated;
            description
              "Enumeration for the
               'TLS_DHE_PSK_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_RSA_PSK_WITH_CAMELLIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_RSA_PSK_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_RSA_PSK_WITH_CAMELLIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_RSA_PSK_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_256_CBC_SHA384'
               algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
```

```
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_RSA_PSK_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_RSA_PSK_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
```

```
            Addition of the Camellia Cipher Suites to Transport
            Layer Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_AES_128_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_RSA_WITH_AES_256_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_128_CCM {
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_256_CCM {
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
```

```
            AES-CCM Cipher Suites for Transport Layer Security
            (TLS)";
        }
        enum TLS_RSA_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_RSA_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_128_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_128_CCM'
             algorithm.";
```

```
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_256_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_PSK_WITH_AES_128_CCM {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_PSK_WITH_AES_256_CCM {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_256_CCM_8'
             algorithm.";
```

```
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_DHE_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_DHE_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_DHE_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_DHE_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECCPWD_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECCPWD_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECCPWD_WITH_AES_128_CCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_128_CCM_SHA256'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECCPWD_WITH_AES_256_CCM_SHA384 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_256_CCM_SHA384'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_SHA256_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_SHA256_SHA256' algorithm.";
          reference
            "RFC 9150:
               TLS 1.3 Authentication and Integrity-Only Cipher
               Suites";
        }
        enum TLS_SHA384_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_SHA384_SHA384' algorithm.";
          reference
            "RFC 9150:
               TLS 1.3 Authentication and Integrity-Only Cipher
               Suites";
        }
        enum TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC'
             algorithm.";
          reference
            "RFC 9189:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.2";
        }
        enum TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC' algorithm.";
          reference
            "RFC 9189:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.2";
        }
```

```
        enum TLS_GOSTR341112_256_WITH_28147_CNT_IMIT {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_28147_CNT_IMIT' algorithm.";
          reference
            "RFC 9189:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.2";
        }
        enum TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_L {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_L' algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.3";
        }
        enum TLS_GOSTR341112_256_WITH_MAGMA_MGM_L {
          status deprecated;
          description
            "Enumeration for the 'TLS_GOSTR341112_256_WITH_MAGMA_MGM_L'
             algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.3";
        }
        enum TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_S {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_S' algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.3";
        }
        enum TLS_GOSTR341112_256_WITH_MAGMA_MGM_S {
          status deprecated;
          description
            "Enumeration for the 'TLS_GOSTR341112_256_WITH_MAGMA_MGM_S'
             algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
```

```
            Protocol Version 1.3";
        }
        enum TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256'
             algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
```

```
        }
        enum TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 8442:
               ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
               1.2 and DTLS 1.2";
        }
        enum TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 8442:
               ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
               1.2 and DTLS 1.2";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_CCM_8_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_128_CCM_8_SHA256' algorithm.";
          reference
            "RFC 8442:
               ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
               1.2 and DTLS 1.2";
        }
```

```
          enum TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256 {
            description
              "Enumeration for the
               'TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256' algorithm.";
            reference
              "RFC 8442:
                 ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
                 1.2 and DTLS 1.2";
          }
        }
        description
          "An enumeration for TLS cipher-suite algorithms.";
      }

    }

    <CODE ENDS>
```

Appendix B. Change Log

B.1. 00 to 01

* Noted that '0.0.0.0' and '::' might have special meanings.

* Renamed "keychain" to "keystore".

B.2. 01 to 02

* Removed the groupings containing transport-level configuration. Now modules contain only the transport-independent groupings.

* Filled in previously incomplete 'ietf-tls-client' module.

* Added cipher suites for various algorithms into new 'ietf-tls-common' module.

B.3. 02 to 03

* Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

* Fixed description statement for leaf 'trusted-ca-certs'.

B.4. 03 to 04

* Updated title to "YANG Groupings for TLS Clients and TLS Servers"

* Updated leafref paths to point to new keystore path

* Changed the YANG prefix for ietf-tls-common from 'tlscom' to 'tlscmn'.

* Added TLS protocol verions 1.0 and 1.1.

* Made author lists consistent

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated YANG to use typedefs around leafrefs to common keystore paths

* Now inlines key and certificates (no longer a leafref to keystore)

B.5. 04 to 05

* Merged changes from co-author.

B.6. 05 to 06

* Updated to use trust anchors from trust-anchors draft (was keystore draft)

* Now Uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

B.7. 06 to 07

* factored the tls-[client|server]-groupings into more reusable groupings.

* added if-feature statements for the new "x509-certificates" feature defined in draft-ietf-netconf-trust-anchors.

B.8. 07 to 08

* Added a number of compatibility matrices to Section 5 (thanks Frank!)

* Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

B.9. 08 to 09

* Updated examples to reflect update to groupings defined in the keystore draft.

* Add TLS keepalives features and groupings.

* Prefixed top-level TLS grouping nodes with 'tls-' and support mashups.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

* Reformatted the YANG modules.

B.11. 10 to 11

* Collapsed all the inner groupings into the top-level grouping.

* Added a top-level "demux container" inside the top-level grouping.

* Added NACM statements and updated the Security Considerations section.

* Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

B.12. 11 to 12

* In server model, made 'client-authentication' a 'presence' node indicating that the server supports client authentication.

* In the server model, added a 'required-or-optional' choice to 'client-authentication' to better support protocols such as RESTCONF.

* In the server model, added a 'inline-or-external' choice to 'client-authentication' to better support consuming data models that prefer to keep client auth with client definitions than in a model principally concerned with the "transport".

* In both models, removed the "demux containers", floating the nacm:default-deny-write to each descendant node, and adding a note to model designers regarding the potential need to add their own demux containers.

* Fixed a couple references (section 2 --> section 3)

B.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

B.14. 12 to 13

* Removed 'container' under 'client-identity' to match server model.

* Updated examples to reflect change grouping in keystore module.

B.15. 13 to 14

* Removed the "certificate" container from "client-identity" in the ietf-tls-client module.

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

B.16. 14 to 15

* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:certificates-ref" to a container that uses "ts:inline-or-truststore-certs-grouping".

B.17. 15 to 16

* Removed unnecessary if-feature statements in the -client and -server modules.

* Cleaned up some description statements in the -client and -server modules.

* Fixed a canonical ordering issue in ietf-tls-common detected by new pyang.

B.18. 16 to 17

* Removed choice inline-or-external by removing the 'external' case and flattening the 'local' case and adding a "client-auth-supported" feature.

* Removed choice required-or-optional.

* Updated examples to include the "*-key-format" nodes.

* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:tls-public-key-format" (must expr for ref'ed keys are TBD).

B.19. 17 to 18

* Removed the unused "external-client-auth-supported" feature.

* Made client-indentity optional, as there may be over-the-top auth instead.

* Added augment to uses of inline-or-keystore-symmetric-key-grouping for a psk "id" node.

* Added missing presence container "psks" to ietf-tls-server's "client-authentication" container.

* Updated examples to reflect new "bag" addition to truststore.

* Removed feature-limited caseless 'case' statements to improve tree diagram rendering.

* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.

* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags).

* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

B.20. 18 to 19

* Updated the "keepalives" containers in part to address Michal Vasko's request to align with RFC 8071, and in part to better align to RFC 6520.

* Removed algorithm-mapping tables from the "TLS Common Model" section

* Removed the 'algorithm' node from the examples.

* Renamed both "client-certs" and "server-certs" to "ee-certs"

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

B.21. 19 to 20

* Modified the 'must' expression in the "ietf-tls-client:server-authention" node to cover the "raw-public-keys" and "psks" nodes also.

* Added a "must 'ca-certs or ee-certs or raw-public-keys or psks'" statement to the ietf-tls-server:client-authentication" node.

* Added "mandatory true" to "choice auth-type" and a "presence" statement to its ancestor.

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

* Moved the "ietf-tls-common" module section to proceed the other two module sections.

* Updated the Security Considerations section.

B.22. 20 to 21

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

B.23. 21 to 22

* In both the "client-authentication" and "server-authentication" subtrees, replaced the "psks" node from being a P-container to a leaf of type "empty".

* Cleaned up examples (e.g., removed FIXMEs)

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

* Updated the "psk" sections in the "ietf-tls-client" and "ietf-tls-server" modules to more correctly reflect RFC 4279.

B.24. 22 to 23

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

B.25. 23 to 24

* Added missing reference to "FIPS PUB 180-4".

* Added identity "tls-1.3" and updated description statement in other identities indicating that the protocol version is obsolete and enabling the feature is NOT RECOMMENDED.

* Added XML-comment above examples explaining the reason for the unexpected top-most element's presence.

* Added missing "client-ident-raw-public-key" and "client-ident-psk" featutes.

* Aligned modules with 'pyang -f' formatting.

* Fixed nits found by YANG Doctor reviews.

* Added a 'Contributors' section.

B.26. 24 to 25

* Added TLS 1.3 references.

* Clarified support for various TLS protocol versions.

* Moved algorithms in ietf-tls-common (plus more) to IANA-maintained modules

* Added "config false" lists for algorithms supported by the server.

* Fixed issues found during YANG Doctor review.

B.27. 25 to 26

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

* Minor editorial nits

B.28. 26 to 27

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

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

* Created identityref-based typedef for the IANA alg identity base.

* Major update to support TLS 1.3.

B.29. 27 to 28

* Fixed draft text to refer to new "identity" values (e.g., s/tls-1.3/tls13).

* Added ietf-tls-common:generate-public-key() RPC.

B.30. 28 to 29

* Updated modules to IANA-maintained module in Appendix A to 2022-06-16.

B.31. 29 to 30

* Fixed 'must' expressions.

* Added missing 'revision' statement.

B.32. 30 to 31

* Updated per Shepherd reviews impacting the suite of drafts.

B.33. 31 to 32

* Updated per Shepherd reviews impacting the suite of drafts.

B.34. 32 to 33

* Updated per Tom Petch review.

* Added RPC-reply to 'generate-public-key" RPC example.

B.35. 33 to 34

* Addresses AD review comments.

* Added note to Editor to fix line foldings.

* Introduction now more clearly identifies the "ietf-" and "iana-" modules defined.

* Clarified that the modules, when implemented, do not define any protocol-accessible nodes.

* Clarified that IANA may deprecate and/or obsolete identities over time.

* Added Security Consideration for the "generate-public-key" RPC.

* Added Security Considerations text to also look a SC-section from imported modules.

* Added missing if-feature statements.

* Fixed private-key "must" expressions to not require public-key nodes to be present.

* Fixed ident-tls12-psk and ident-tls13-psk YANG and references.

* Renamed leaf from "bits" to "num-bits".

* Added missing "ordered-by user" statement.

* Added container "private-key-encoding" to wrap existing choice.

* Renamed container "encrypt-with" to "encrypted".

* Renamed leaf from "hide" to "hidden".

* Removed "public-key-format" and "public-key" nodes from examples.

B.36. 34 to 35

* Addresses AD review by Rob Wilton.

B.37. 35 to 36

* Complete tls10/tls11 removal and update Jeff’s email.

B.38. 36 to 37

* Addresses 1st-round of IESG reviews.

B.39. 37 to 39

* Addresses issues found in OpsDir review of the ssh-client-server draft.

* Replaced identities with enums in the IANA module.

* Add refs to where the ’operational’ and ’system’ datastores are defined.

* Updated Introduction to read more like the Abstract

* Updated Editor-notes to NOT remove the script (just remove the initial IANA module)

* Renamed Security Considerations section s/Template for/ Considerations for/

* s/defines/presents/ in a few places.

* Renamed script from 'gen-identities.py' to 'gen-yang-module.py'

* Removed the removeInRFC="true" attribute in Appendix sections

B.40. 39 to 40

* Address IESG review comments.

B.41. 40 to 41

* Updated to reflect comments from Paul Wouters.

* Fixed the "generate-asymmetric-key-pair" RPC to return the location to where hidden keys are created.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balázs Kovács, Benoit Claise, Bert Wijnen, David Lamparter, Dhruv Dhody, Éric Vyncke, Gary Wu, Henk Birkholz, Jeff Hartley, Jürgen Schönwälder, Ladislav Lhotka, Liang Xia, Martin Björklund, Martin Thomson, Mehmet Ersue, Michal Vako, Murray Kucherawy, Paul Wouters, Phil Shafer, Qin Wu, Radek Krejci, Rob Wilton, Roman Danyliw, Russ Housley, Sean Turner, Tom Petch, and Thomas Martin.

Contributors

Special acknowledgement goes to Gary Wu who contributed the "ietf-tls-common" module, and Tom Petch who carefully ensured that references were set correctly throughout.

Author's Address

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

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

K. Watsen
Watsen Networks

10 July 2020

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

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

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

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

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

* Appendix A. Change Log

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

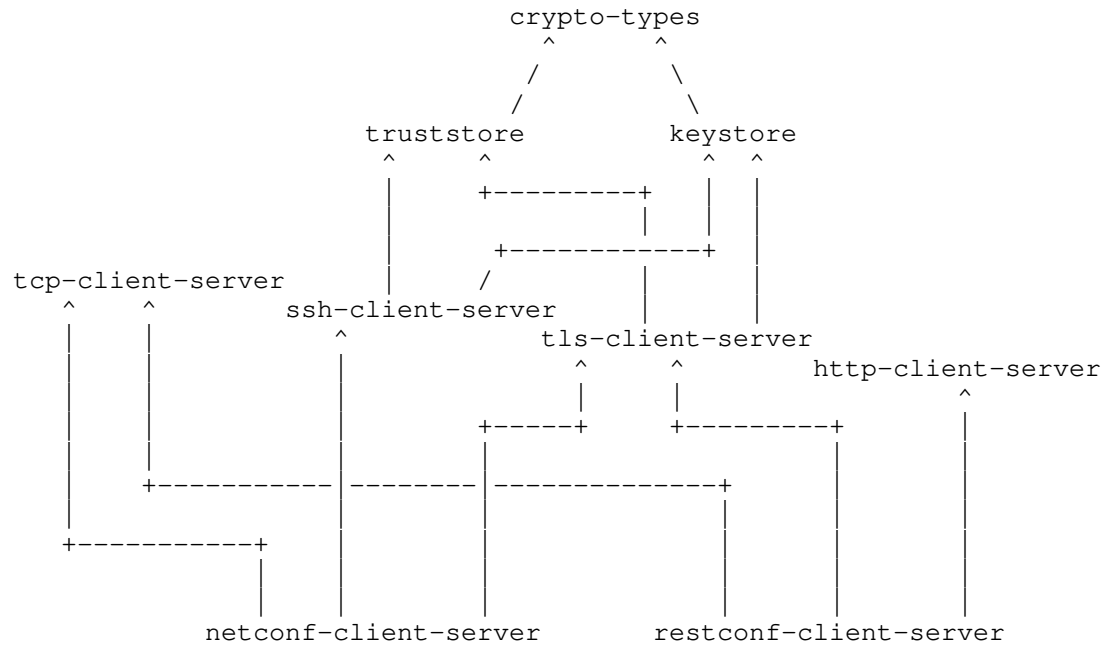
This document defines a YANG 1.1 [RFC7950] data model for configuring globally-accessible bags of certificates and public keys that can be referenced by other data models for trust.

1.1. Relation to other RFCs

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

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

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



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

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

2. The "ietf-truststore" Module

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

2.1. Data Model Overview

2.1.1. Features

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

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

2.1.2. Typedefs

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

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

Comments:

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

* The leafrefs refer to certificates, public keys, and bags. These typedefs are provided primarily as an aid to downstream modules that import the "ietf-truststore" module.

2.1.3. Groupings

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

```
Groupings:
  +-- local-or-truststore-certs-grouping
  +-- local-or-truststore-public-keys-grouping
  +-- truststore-grouping
```

Each of these groupings are presented in the following subsections.

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

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

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

Comments:

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

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

* For the "local-definition" option, the "certificate" node uses the "trust-anchor-cert-grouping" grouping discussed in Section 2.1.3.6 of [I-D.ietf-netconf-crypto-types].

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

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

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

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

Comments:

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

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

* For the "local-definition" option, the "public-key" node uses the "public-key-grouping" grouping discussed in Section 2.1.3.3 of [I-D.ietf-netconf-crypto-types].

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

2.1.3.3. The "truststore-grouping" Grouping

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

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

Comments:

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

* For certificates, each certificate is defined by the "trust-anchor-cert-grouping" grouping Section 2.1.3.6 of [I-D.ietf-netconf-crypto-types]. Thus the "cert-data" node is a CMS structure that can be composed of a chain of one or more certificates. Additionally, the "certificate-expiration" notification enables the server to alert clients when certificates are nearing or have already expired.

* For public keys, each public key is defined by the "public-key-grouping" grouping Section 2.1.3.3 of [I-D.ietf-netconf-crypto-types]. Thus the "public-key" node can be one of any number of structures specified by the "public-key-format" identity node.

2.1.4. Protocol-accessible Nodes

The following diagram lists all the protocol-accessible nodes defined in the "ietf-truststore" module:

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

Comments:

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

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

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

2.2. Example Usage

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

2.2.1. A Truststore Instance

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

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

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

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

Following is the full example:

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

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

    <!-- CA Certs for Authenticating Servers Using Private PKIs -->
    <certificate-bag>
      <name>trusted-server-ca-certs</name>
      <description>
        Trust anchors (i.e. CA certs) used to authenticate server
        certificates.  A server certificate is authenticated if its
        end-entity certificate has a chain of trust to one of these
        certificates.
      </description>
      <certificate>
        <name>Server Cert Issuer #1</name>
        <cert-data>base64encodedvalue==</cert-data>
      </certificate>
      <certificate>
        <name>Server Cert Issuer #2</name>
        <cert-data>base64encodedvalue==</cert-data>
      </certificate>
    </certificate-bag>

    <!-- End Entity Certs for Authenticating Servers -->
    <certificate-bag>
      <name>trusted-server-ee-certs</name>
```

```
          <description>
            Specific end-entity certificates used to authenticate server
            certificates.  A server certificate is authenticated if its
            end-entity certificate is an exact match to one of these
            certificates.
          </description>
          <certificate>
            <name>My Application #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>My Application #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </certificate-bag>

        <!-- CA Certs for Authenticating Clients -->
        <certificate-bag>
          <name>trusted-client-ca-certs</name>
          <description>
            Trust anchors (i.e. CA certs) used to authenticate client
            certificates.  A client certificate is authenticated if its
            end-entity certificate has a chain of trust to one of these
            certificates.
          </description>
          <certificate>
            <name>Client Identity Issuer #1</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Client Identity Issuer #2</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </certificate-bag>

        <!-- Entity Certs for Authenticating Clients -->
        <certificate-bag>
          <name>trusted-client-ee-certs</name>
          <description>
            Specific end-entity certificates used to authenticate client
            certificates.  A client certificate is authenticated if its
            end-entity certificate is an exact match to one of these
            certificates.
          </description>
          <certificate>
            <name>George Jetson</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
```

```
        <certificate>
          <name>Fred Flintstone</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificate-bag>
    </certificate-bags>

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

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

          This list of SSH public keys is analogous to OpenSSH's
          "/etc/ssh/ssh_known_hosts" file.
        </description>
        <public-key>
          <name>corp-fw1</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>corp-fw2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
      </public-key-bag>

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

```
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>Application Instance #2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
      </public-key-bag>

      <!-- Raw Public Keys for TLS Servers -->
      <public-key-bag>
        <name>Raw Public Keys for TLS Servers</name>
        <public-key>
          <name>Raw Public Key #1</name>
          <public-key-format>
            ct:subject-public-key-info-format</public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>Raw Public Key #2</name>
          <public-key-format>
            ct:subject-public-key-info-format
          </public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
      </public-key-bag>

      <!-- Raw Public Keys for TLS Clients -->
      <public-key-bag>
        <name>Raw Public Keys for TLS Clients</name>
        <public-key>
          <name>Raw Public Key #1</name>
          <public-key-format>
            ct:subject-public-key-info-format
          </public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
        <public-key>
          <name>Raw Public Key #2</name>
          <public-key-format>
            ct:subject-public-key-info-format
        </public-key-format>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
```

```
      </public-key-bag>
    </public-key-bags>
  </truststore>
```

2.2.2. A Certificate Expiration Notification

The following example illustrates the "certificate-expiration" notification (per Section 2.1.3.4 of [I-D.ietf-netconf-crypto-types]) for a certificate configured in the truststore in Section 2.2.1.

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

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

2.2.3. The "Local or Truststore" Groupings

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

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

```
module ex-truststore-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-truststore-usage";
  prefix "etu";

  import ietf-truststore {
```

```
      prefix ts;
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

    organization
     "Example Corporation";

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

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

    revision "2020-07-10" {
      description
       "Initial version";
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

    container truststore-usage {
      description
        "An illustration of the various truststore groupings.";

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

      list public-key {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this cert.";
        }
        uses ts:local-or-truststore-public-keys-grouping;
        description
```

```
          "An public key that may be configured locally or be
           a reference to a public key in the truststore.";
      }
    }
  }
```

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

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

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

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

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

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

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

     <cert>
       <name>example 1b</name>
       <local-definition>
         <name>my-trusted-client-ca-certs</name>
         <certificate>
           <name>Client Identity Issuer #1</name>
           <cert>base64encodedvalue==</cert>
         </certificate>
         <certificate>
           <name>Client Identity Issuer #2</name>
           <cert>base64encodedvalue==</cert>
         </certificate>
       </local-definition>
     </cert>


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

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

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

```
            </public-key-format>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
          <public-key>
            <name>corp-fw2</name>
            <public-key-format>
              ct:ssh-public-key-format
            </public-key-format>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
        </local-definition>
      </public-key>

   </truststore-usage>
```

2.3. YANG Module

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

<CODE BEGINS> file "ietf-truststore@2020-07-10.yang"

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

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

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

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

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

  description
```

```
      "This module defines a Truststore to centralize management
       of trust anchors including certificates and public keys.

       Copyright (c) 2020 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

       Redistribution and use in source and binary forms, with
       or without modification, is permitted pursuant to, and
       subject to the license terms contained in, the Simplified
       BSD License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC BBBB
       (https://www.rfc-editor.org/info/rfcBBBB); see the RFC
       itself for full legal notices.

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

    revision 2020-07-10 {
      description
        "Initial version";
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

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

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

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

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

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

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

  typedef certificate-bag-ref {
    type leafref {
      path "/ts:truststore/ts:certificate-bags/"
           + "ts:certificate-bag/ts:name";
    }
    description
      "This typedef defines a reference to a certificate bag
       defined in the Truststore.";
  }

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

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

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


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

  grouping local-or-truststore-certs-grouping {
    description
      "A grouping that allows the certificates to be either
       configured locally, within the using data model, or be a
       reference to a certificate bag stored in the Truststore.";
    choice local-or-truststore {
      nacm:default-deny-write;
      mandatory true;
      case local {
        if-feature "local-definitions-supported";
        container local-definition {
          description
            "A container for locally configured trust anchor
             certificates.";
          list certificate {
            key "name";
            min-elements 1;
            description
              "A trust anchor certificate.";
            leaf name {
              type string;
              description
                "An arbitrary name for this certificate.";
            }
            uses ct:trust-anchor-cert-grouping {
              refine "cert-data" {
                mandatory true;
              }
            }
```

```
            }
          }
        }
        case truststore {
          if-feature "truststore-supported";
          if-feature "certificates";
          leaf truststore-reference {
            type ts:certificate-bag-ref;
            description
              "A reference to a certificate bag that exists in the
               Truststore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the Truststore.";
      }
    }


    grouping local-or-truststore-public-keys-grouping {
      description
        "A grouping that allows the public keys to be either
         configured locally, within the using data model, or be a
         reference to a public key bag stored in the Truststore.";
      choice local-or-truststore {
        nacm:default-deny-write;
        mandatory true;
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "Container to hold local public key definitions.";
            list public-key {
              key name;
              description
                "A public key definition.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this public key.";
              }
              uses ct:public-key-grouping;
            }
          }
        }
        case truststore {
          if-feature "truststore-supported";
```

```
          if-feature "public-keys";
          leaf truststore-reference {
            type ts:public-key-bag-ref;
            description
              "A reference to a bag of public keys that exist
               in the Truststore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the Truststore.";
      }
    }

    grouping truststore-grouping {
      description
        "Grouping definition enables use in other contexts.  Where
         used, implementations SHOULD augment new 'case' statements
         into the local-or-truststore 'choice' statements to supply
         leafrefs to the model-specific location.";
      container certificate-bags {
        nacm:default-deny-write;
        if-feature "certificates";
        presence
          "Indicates that certificate bags have been configured.";
        description
          "A collection of certificate bags.";
        list certificate-bag {
          key "name";
          min-elements 1;
          description
            "A bag of certificates.  Each bag of certificates SHOULD
             be for a specific purpose.  For instance, one bag could
             be used to authenticate a specific set of servers, while
             another could be used to authenticate a specific set of
             clients.";
          leaf name {
            type string;
            description
              "An arbitrary name for this bag of certificates.";
          }
          leaf description {
            type string;
            description
              "A description for this bag of certificates.  The
              intended purpose for the bag SHOULD be described.";
          }
          list certificate {
```

```
            key "name";
            min-elements 1;
            description
              "A trust anchor certificate.";
            leaf name {
              type string;
              description
                "An arbitrary name for this certificate.";
            }
            uses ct:trust-anchor-cert-grouping {
              refine "cert-data" {
                mandatory true;
              }
            }
          }
        }
      }
      container public-key-bags {
        nacm:default-deny-write;
        if-feature "public-keys";
        presence
          "Indicates that public keys have been configured.";
        description
          "A collection of public key bags.";
        list public-key-bag {
          key "name";
          min-elements 1;
          description
            "A bag of public keys.  Each bag of keys SHOULD be for
             a specific purpose.  For instance, one bag could be used
             authenticate a specific set of servers, while another
             could be used to authenticate a specific set of clients.";
          leaf name {
            type string;
            description
              "An arbitrary name for this bag of public keys.";
          }
          leaf description {
            type string;
            description
              "A description for this bag public keys.  The
               intended purpose for the bag SHOULD be described.";
          }
          list public-key {
            key "name";
            min-elements 1;
            description
              "A public key.";
```

```
            leaf name {
              type string;
              description
                "An arbitrary name for this public key.";
            }
            uses ct:public-key-grouping;
          }
        }
      }
    }

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

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

<CODE ENDS>

3. Support for Built-in Trust Anchors

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

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

As built-in trust anchors are provided by the system, they are present in <operational>. The example below illustrates what the Truststore in <operational> might look like for a server in its factory default state.

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

       <certificate-bag or:origin="or:system">
         <name>Built-In Manufacturer CA Certificates</name>
         <description>
           Certificates built into the device for authenticating
           manufacturer-signed objects, such as TLS server certificates,
           vouchers, etc.
         </description>
         <certificate>
           <name>Manufacturer Root CA Cert</name>
           <cert-data>base64encodedvalue==</cert-data>
         </certificate>
       </certificate-bag>

       <certificate-bag or:origin="or:system">
         <name>Built-In Public CA Certificates</name>
         <description>
           Certificates built into the device for authenticating
           certificates issued by public certificate authorities,
           such as the end-entity certificate for web servers.
         </description>
         <certificate>
           <name>Public Root CA Cert 1</name>
           <cert-data>base64encodedvalue==</cert-data>
         </certificate>
         <certificate>
           <name>Public Root CA Cert 2</name>
           <cert-data>base64encodedvalue==</cert-data>
         </certificate>
         <certificate>
           <name>Public Root CA Cert 3</name>
           <cert-data>base64encodedvalue==</cert-data>
         </certificate>
       </certificate-bag>

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

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

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

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

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

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

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

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

        Only the subset of the certificates that are referenced
        by other configuration nodes need to be copied.  For
        instance, only "Public Root CA Cert 3" is present here.

        No new certificates can be added, nor existing certificate
        values changed.  Missing certificates have no effect on
        "operational" when the configuration is applied.
      </description>
      <certificate>
        <name>Public Root CA Cert 3</name>
        <cert-data>base64encodedvalue==</cert-data>
      </certificate>
    </certificate-bag>

  </certificate-bags>
</truststore>
```

4. Security Considerations

4.1. Data at Rest

The YANG module defined in this document defines a mechanism called a "truststore" that, by its name, suggests that it will protect its contents from unauthorized modification.

Security controls for the API (i.e., data in motion) are discussed in Section 4.2, but controls for the data at rest cannot be specified by the YANG module.

In order to satisfy the expectations of a "truststore", it is RECOMMENDED that implementations ensure that the truststore contents are signed when persisted to non-volatile memory, to prevent unauthorized modifications from being made undetected.

4.2. The "ietf-truststore" YANG Module

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

All of the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "truststore" instance, may be considered sensitive or vulnerable in some network environments. For instance, any modification to a trust anchor or reference to a trust anchor may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-truststore
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

5.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registration is requested:

```
      name:         ietf-truststore
      namespace:    urn:ietf:params:xml:ns:yang:ietf-truststore
      prefix:       ts
      reference:    RFC BBBB
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

6.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-17, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-17>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-19>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-19>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Added features "x509-certificates" and "ssh-host-keys".

* Added nacm:default-deny-write to "trust-anchors" container.

A.2. 01 to 02

* Switched "list pinned-certificate" to use the "trust-anchor-cert-grouping" from crypto-types. Effectively the same definition as before.

A.3. 02 to 03

* Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

A.4. 03 to 04

* Added groupings 'local-or-truststore-certs-grouping' and 'local-or-truststore-host-keys-grouping', matching similar definitions in the keystore draft. Note new (and incomplete) "truststore" usage!

* Related to above, also added features 'truststore-supported' and 'local-trust-anchors-supported'.

A.5. 04 to 05

* Renamed "trust-anchors" to "truststore"

* Removed "pinned." prefix everywhere, to match truststore rename

* Moved everything under a top-level 'grouping' to enable use in other contexts.

* Renamed feature from 'local-trust-anchors-supported' to 'local-definitions-supported' (same name used in keystore)

* Removed the "require-instance false" statement from the "*-ref" typedefs.

* Added missing "ssh-host-keys" and "x509-certificates" if-feature statements

A.6. 05 to 06

* Editorial changes only.

A.7. 06 to 07

* Added Henk Birkholz as a co-author (thanks Henk!)

* Added PSKs and raw public keys to Truststore.

A.8. 07 to 08

* Added new "Support for Built-in Trust Anchors" section.

* Removed spurious "uses ct:trust-anchor-certs-grouping" line.

* Removed PSK from model.

A.9. 08 to 09

* Removed remaining PSK references from text.
* Wrapped each top-level list with a container.
* Introduced "bag" term.
* Merged "SSH Public Keys" and "Raw Public Keys" in a single "Public Keys" bag. Consuming downstream modules (i.e., "ietf-[ssh/tls]-[client/server]) refine the "public-key-format" to be either SSH or TLS specific as needed.

A.10. 09 to 10

* Removed "algorithm" node from examples.
* Removed the no longer used statements supporting the old "ssh-public-key" and "raw-public-key" nodes.
* Added a "Note to Reviewers" note to first page.

A.11. 10 to 11

* Corrected module prefix registered in the IANA Considerations section.
* Modified 'local-or-truststore-certs-grouping' to use a list (not a leaf-list).
* Added new example section "The Local or Truststore Groupings".
* Clarified expected behavior for "built-in" certificates in <operational>
* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
* Updated the Security Considerations section.

A.12. 11 to 12

* Fixed a copy/paste issue in the "Data at Rest" Security Considerations section.

Acknowledgements

The authors especially thank Henk Birkholz for contributing YANG to the ietf-truststore module supporting raw public keys and PSKs (pre-shared or pairwise-symmetric keys). While these contributions were eventually replaced by reusing the existing support for asymmetric and symmetric trust anchors, respectively, it was only thru Henk's initiative that the WG was able to come to that result.

The authors additionally thank the following for helping give shape to this work (ordered by first name): Balazs Kovacs, Eric Voit, Juergen Schoenwaelder, Liang Xia, Martin Bjorklund, and Nick Hancock.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-28

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.

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:

* 2024-03-16 --> the publication date of this draft

The "Relation to other RFCs" section Section 1.1 contains the text "one or more YANG modules" and, later, "modules". This text is sourced from a file in a context where it is unknown how many modules a draft defines. The text is not wrong as is, but it may be improved by stating more directly how many modules are defined.

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

The following Appendix section is to be removed prior to publication:

* Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on 17 September 2024.

Copyright Notice

Table of Contents

1. Introduction

This document presents a YANG 1.1 [RFC7950] module having the following characteristics:

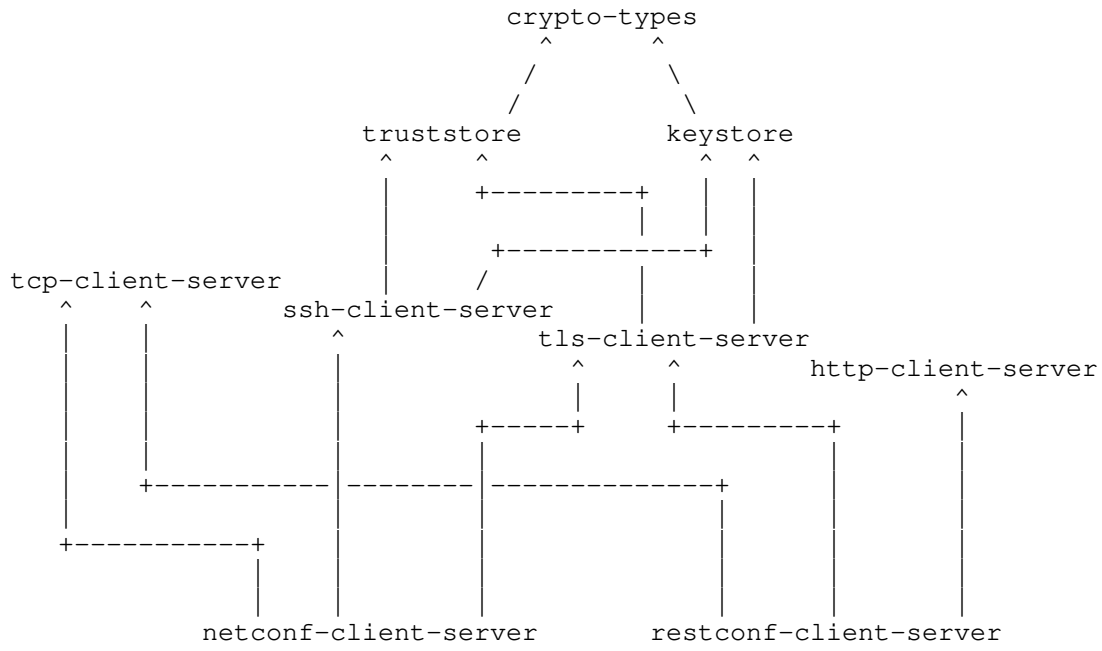
- Provide a central truststore for storing raw public keys and/or certificates.
- Provide support for storing named bags of raw public keys and/or named bags of certificates.
- Provide types that can be used to reference raw public keys or certificates stored in the central truststore.
- Provide groupings that enable raw public keys and certificates to be configured 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 one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, support the configuration of both the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The dependency relationship between the primary YANG groupings defined in the various RFCs is presented in the below diagram. In some cases, a draft may define secondary groupings that introduce dependencies not illustrated in the diagram. The labels in the diagram are a shorthand name for the defining RFC. The citation reference for shorthand name is 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.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label 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 in [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" to the word "truststore" to refer to the top-level instance of the "truststore-grouping", when the "central-truststore-supported" feature is enabled. Please be aware that consuming YANG modules MAY instantiate the "truststore-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 Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-truststore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-truststore" module:

```
Features:
  +-- central-truststore-supported
  +-- 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 as an aid to 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 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 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 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 2.1.4.7 of [I-D.ietf-netconf-crypto-types].

* 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 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 [I-D.ietf-netconf-crypto-types].

* 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 structure supporting certificates and public keys is essentially the same, having an outer list of "bags" containing an inner list of objects (certificates or public keys). The bags enable trust anchors serving a common purpose to be grouped and referenced together.

* For certificates, each certificate is defined by the "trust-anchor-cert-grouping" grouping Section 2.1.4.7 of [I-D.ietf-netconf-crypto-types]. The "cert-data" node is a CMS structure that can be composed of a chain of one or more certificates. Additionally, the "certificate-expiration" notification enables the server to alert clients when certificates are nearing or have already expired.

* For public keys, each public key is defined by the "public-key-grouping" grouping Section 2.1.4.4 of [I-D.ietf-netconf-crypto-types]. 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 an instance of the "truststore-grouping" grouping discussed in Section 2.1.3.5.

* The top-level node "truststore" is additionally constrained by the feature "central-truststore-supported".

* The "truststore-grouping" grouping is discussed in Section 2.1.3.5.

* The reason for why the "truststore-grouping" exists separate from the protocol-accessible nodes definition is to enable instances of the truststore to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The examples in this section are encoded using XML, such as might be the case when using the NETCONF protocol. Other encodings MAY be used, such as JSON when using the RESTCONF protocol.

2.2.1. A Truststore Instance

This section presents an example illustrating trust anchors in <intended>, as per Section 2.1.4. Please see Section 3 for an example illustrating built-in values in <operational>.

The example contained in this section defines eight bags of trust anchors. There are four certificate-based bags and four public key based bags. The following diagram provides an overview of the contents in the example:

```
Certificate Bags
  +-- Trust anchor certs for authenticating a set of remote servers
  +-- End entity certs for authenticating a set of remote servers
  +-- Trust anchor certs for authenticating a set of remote clients
  +-- End entity certs for authenticating a set of remote clients

Public Key Bags
  +-- SSH keys to authenticate a set of remote SSH server
  +-- SSH keys to authenticate a set of remote SSH clients
  +-- Raw public keys to authenticate a set of remote SSH server
  +-- Raw public keys to authenticate a set of remote SSH clients
```

Following is the full example:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<truststore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- A bag of Certificate Bags -->
  <certificate-bags>

    <!-- Trust Anchor Certs for Authenticating Servers -->
    <certificate-bag>
      <name>trusted-server-ca-certs</name>
```

```
        <description>
          Trust anchors (i.e. CA certs) used to authenticate server
          certificates.  A server certificate is authenticated if its
          end-entity certificate has a chain of trust to one of these
          certificates.
        </description>
        <certificate>
          <name>Server Cert Issuer #1</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Server Cert Issuer #2</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificate-bag>

      <!-- End Entity Certs for Authenticating Servers -->
      <certificate-bag>
        <name>trusted-server-ee-certs</name>
        <description>
          Specific end-entity certificates used to authenticate server
          certificates.  A server certificate is authenticated if its
          end-entity certificate is an exact match to one of these
          certificates.
        </description>
        <certificate>
          <name>My Application #1</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>My Application #2</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificate-bag>

      <!-- Trust Anchor Certs for Authenticating Clients -->
      <certificate-bag>
        <name>trusted-client-ca-certs</name>
        <description>
          Trust anchors (i.e. CA certs) used to authenticate client
          certificates.  A client certificate is authenticated if its
          end-entity certificate has a chain of trust to one of these
          certificates.
        </description>
        <certificate>
          <name>Client Identity Issuer #1</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
```

```
          <certificate>
            <name>Client Identity Issuer #2</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </certificate-bag>

        <!-- End Entity Certs for Authenticating Clients -->
        <certificate-bag>
          <name>trusted-client-ee-certs</name>
          <description>
            Specific end-entity certificates used to authenticate client
            certificates.  A client certificate is authenticated if its
            end-entity certificate is an exact match to one of these
            certificates.
          </description>
          <certificate>
            <name>George Jetson</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
          <certificate>
            <name>Fred Flintstone</name>
            <cert-data>BASE64VALUE=</cert-data>
          </certificate>
        </certificate-bag>
      </certificate-bags>

      <!-- A List of Public Key Bags -->
      <public-key-bags>

        <!-- Public Keys for Authenticating SSH Servers -->
        <public-key-bag>
          <name>trusted-ssh-public-keys</name>
          <description>
            Specific SSH public keys used to authenticate SSH server
            public keys.  An SSH server public key is authenticated if
            its public key is an exact match to one of these public keys.

            This list of SSH public keys is analogous to OpenSSH's
            "/etc/ssh/ssh_known_hosts" file.
          </description>
          <public-key>
            <name>corp-fw1</name>
            <public-key-format>ct:ssh-public-key-format</public-key-form\
   at>
            <public-key>BASE64VALUE=</public-key>
          </public-key>
          <public-key>
            <name>corp-fw2</name>
```

```
          <public-key-format>ct:ssh-public-key-format</public-key-form\
   at>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>

      <!-- SSH Public Keys for Authenticating Application A -->
      <public-key-bag>
        <name>SSH Public Keys for Application A</name>
        <description>
          SSH public keys used to authenticate application A's SSH
          public keys.  An SSH public key is authenticated if it
          is an exact match to one of these public keys.
        </description>
        <public-key>
          <name>Application Instance #1</name>
          <public-key-format>ct:ssh-public-key-format</public-key-form\
   at>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>Application Instance #2</name>
          <public-key-format>ct:ssh-public-key-format</public-key-form\
   at>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>

      <!-- Raw Public Keys for TLS Servers -->
      <public-key-bag>
        <name>Raw Public Keys for TLS Servers</name>
        <public-key>
          <name>Raw Public Key #1</name>
          <public-key-format>ct:subject-public-key-info-format</public\
   -key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>Raw Public Key #2</name>
          <public-key-format>ct:subject-public-key-info-format</public\
   -key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>

      <!-- Raw Public Keys for TLS Clients -->
      <public-key-bag>
        <name>Raw Public Keys for TLS Clients</name>
```

```
        <public-key>
          <name>Raw Public Key #1</name>
          <public-key-format>ct:subject-public-key-info-format</public\
   -key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>Raw Public Key #2</name>
          <public-key-format>ct:subject-public-key-info-format</public\
   -key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
      </public-key-bag>
    </public-key-bags>
   </truststore>
```

2.2.2. A Certificate Expiration Notification

The following example illustrates the "certificate-expiration" notification (per Section 2.1.4.6 of [I-D.ietf-netconf-crypto-types]) for a certificate configured in the truststore in Section 2.2.1.

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
       <certificate-bags>
         <certificate-bag>
           <name>trusted-client-ee-certs</name>
           <certificate>
             <name>George Jetson</name>
             <certificate-expiration>
               <expiration-date>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" having 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 "ex-truststore-usage" 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>
        <name>my-trusted-client-ca-certs</name>
        <certificate>
          <name>Client Identity Issuer #1</name>
          <cert>BASE64VALUE=</cert>
        </certificate>
        <certificate>
          <name>Client Identity Issuer #2</name>
          <cert>BASE64VALUE=</cert>
        </certificate>
      </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>
        <name>trusted-ssh-public-keys</name>
        <public-key>
          <name>corp-fw1</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
        </public-key>
        <public-key>
          <name>corp-fw2</name>
          <public-key-format>
            ct:ssh-public-key-format
          </public-key-format>
          <public-key>BASE64VALUE=</public-key>
```

```
        </public-key>
      </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 BBBB: 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 BBBB: A YANG Data Model for a Truststore";
    }

    container truststore-usage {
      description
        "An illustration of the various truststore groupings.";
      list cert {
        key "name";
        leaf name {
          type string;
          description
            "An arbitrary name for this cert.";
        }
        uses ts: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 [I-D.ietf-netconf-crypto-types].

<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 AAAA: YANG Data Types and Groupings for Cryptography";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
    "WG Web  : https://datatracker.ietf.org/wg/netconf
     WG List : NETCONF WG list <mailto:netconf@ietf.org>
```

```
      Author  : Kent Watsen <kent+ietf@watsen.net>";

    description
      "This module defines a 'truststore' to centralize management
       of trust anchors including certificates and public keys.

       Copyright (c) 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 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 2024-03-16 {
      description
        "Initial version";
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

    /****************/
    /*   Features   */
    /****************/

    feature central-truststore-supported {
      description
        "The 'central-truststore-supported' feature indicates that
         the server supports the truststore (i.e., implements the
         'ietf-truststore' module).";
    }

    feature 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 typedef
         'central-certificate-bag-ref'.";
    }

    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' that SHOULD have the typedef
         'central-public-key-bag-ref'.";
    }

    /*****************/
    /*   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 certificate 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 be either
          configured locally, within the using data model, 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' 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 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
             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 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., an SZTP [RFC8572] bootstrap server) or public 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> [I-D.ietf-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 defined in this document defines a mechanism called a "truststore" that, by its name, suggests that its contents are protected from unauthorized modification.

Security controls for the API (i.e., data in motion) are discussed in Section 4.3, but controls for the data at rest (e.g., on disk) cannot be specified by the YANG module.

In order to satisfy the expectations of a "truststore", 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, both).

Trust anchors configured via this module are implicitly trusted to validate certification paths that may include any name, be used for any purpose, 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., SSH, TLS) with mutual authentication.

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

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 3.9 of [I-D.ietf-netconf-crypto-types].

All the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "truststore" instance, may be considered sensitive or vulnerable in some network environments. For instance, any modification to a trust anchor or reference to a trust anchor may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any "rpc" or "action" statements, and thus the security considerations for such is not provided here.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-truststore
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.
```

5.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registration is requested:

```
name:         ietf-truststore
namespace:    urn:ietf:params:xml:ns:yang:ietf-truststore
prefix:       ts
reference:    RFC BBBB
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

6.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-05, 21 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[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>.

[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>.

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 'inline-or-truststore-certs-grouping' and 'inline-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 'inline-definitions-supported' (same name used in keystore)

* Removed the "require-instance false" statement from the "*-ref" typedefs.

* Added missing "ssh-host-keys" and "x509-certificates" if-feature statements

A.6. 05 to 06

* Editorial changes only.

A.7. 06 to 07

* Added Henk Birkholz as a co-author (thanks Henk!)

* Added PSKs and raw public keys to truststore.

A.8. 07 to 08

* Added new "Support for Built-in Trust Anchors" section.

* Removed spurious "uses ct:trust-anchor-certs-grouping" line.

* Removed PSK from model.

A.9. 08 to 09

* Removed remaining PSK references from text.

* Wrapped each top-level list with a container.

* Introduced "bag" term.

* Merged "SSH Public Keys" and "Raw Public Keys" in a single "Public Keys" bag. Consuming downstream modules (i.e., "ietf-[ssh/tls]-[client/server]) refine the "public-key-format" to be either SSH or TLS specific as needed.

A.10. 09 to 10

* Removed "algorithm" node from examples.

* Removed the no longer used statements supporting the old "ssh-public-key" and "raw-public-key" nodes.

* Added a "Note to Reviewers" note to first page.

A.11. 10 to 11

* Corrected module prefix registered in the IANA Considerations section.

* Modified 'inline-or-truststore-certs-grouping' to use a list (not a leaf-list).

* Added new example section "The Local or Truststore Groupings".

* Clarified expected behavior for "built-in" certificates in <operational>

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.12. 11 to 12

* Fixed a copy/paste issue in the "Data at Rest" Security Considerations section.

A.13. 12 to 13

* Fixed issues found by the SecDir review of the "keystore" draft.

A.14. 13 to 14

* Added an "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir.

* Addressed comments raised by YANG Doctor.

A.15. 14 to 15

* Added prefixes to 'path' statements per trust-anchors/issues/1

* Renamed feature "truststore-supported" to "central-truststore-supported".

* Associated with above, generally moved text to refer to a "central" truststore.

* Removed two unecessary/unwanted "min-elements 1" and associated "presence" statements.

* Aligned modules with `pyang -f` formatting.

* Fixed nits found by YANG Doctor reviews.

A.16. 15 to 16

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

A.17. 16 to 17

* fixup the 'WG Web' and 'WG List' lines in YANG module(s)

* fixup copyright (i.e., s/Simplified/Revised/) in YANG module(s)

* Added Informative reference to ma-netmod-with-system

A.18. 17 to 18

* Updated Security Considerations section to address comment received from Carl Wallace.

* Fixed examples to not have line-returns around "identity" encodings.

* Fixed a couple tree diagrams to not create diagrams for "groupings" too.

* Added "if-feature central-truststore-supported" to top-level "trustore" container.

A.19. 18 to 19

* Updated per Shepherd reviews impacting the suite of drafts.

A.20. 19 to 20

* Updated per Shepherd reviews impacting the suite of drafts.

A.21. 20 to 21

* Updated (implicitly) per Tom Petch review.

* Updated per AD's review.

* s/local/inline/ in feature names, grouping names, and node names.

* Updated ref from 'ma-netmod-with-system' to 'ietf-netmod-system-config'.

* Removed special handling text for built-in certs

* Updated section on built-in trust anchors to read almost the same as the section in the keystore draft.

A.22. 21 to 22

* Mostly addresses AD review comments.

* Also added typedefs and groupings similar to those created by Alto WG.

* Added note to Editor to fix line foldings.

* Renamed "truststore" to "central truststore" throughout.

* Removed "built-in" section text that overlaps with the "system-config" draft.

* Added "certificate-ref-grouping" and "public-key-ref-grouping"

* Modified typedef certificate-ref's leafref path to NOT prefix "certificate-bag".

* Modified typedef public-key-ref's leafref path to NOT prefix "public-key-bag".

* Added groupings "certificate-ref-grouping" and "public-key-ref-grouping".

A.23. 22 to 23

* Addresses Gen-ART review by Dale Worley.

* Addresses review by Tom Petch.

A.24. 23 to 24

* Addresses 1st-round of IESG reviews.

A.25. 24 to 26

* Addresses issues found in OpsDir review of the ssh-client-server draft.

* Renamed Security Considerations section s/Template for/ Considerations for/

* s/defines/presents/ in a few places.

* Add refs to where the 'operational' and 'system' datastores are defined.

* Improved Security Consideration for 'cert-data' node.

* s/should/SHOULD/ is one place

A.26. 26 to 28

* Nothing changed. Only bumped for automation...

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, Qin Wu, Rob Wilton, Robert Varga, Roman Danyliw, Paul Kyzivat, and Yoav Nir.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Updates: 8572 (if approved)
Intended status: Standards Track
Expires: 12 December 2020

K. Watsen
Watsen Networks
R. Housley
Vigil Security, LLC
S. Turner
sn3rd

10 June 2020

Conveying a Certificate Signing Request (CSR) in a Secure Zero Touch Provisioning (SZTP) Bootstrapping Request
draft-kwatsen-netconf-sztp-csr-01

Abstract

This draft extends the "get-bootstrapping-data" RPC defined in RFC 8572 to include an optional certificate signing request (CSR), enabling a bootstrapping device to additionally obtain an identity certificate (e.g., an LDevID, from IEEE 802.1AR) as part of the "onboarding information" response provided in the RPC-reply.

Editorial Note (To be removed by RFC Editor)

This draft contains many 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:

* "XXXX" --> the assigned numerical RFC value for this draft

* "AAAA" --> the assigned RFC value for I-D.ietf-netconf-crypto-types

Artwork in this document contains a placeholder value for the publication date of this draft. Please apply the following replacement:

* "2020-06-10" --> the publication date of this draft

This document contains references to other drafts in progress, both in the Normative References section, as well as in body text throughout. Please update the following references to reflect their final RFC assignments:

* I-D.ietf-netconf-crypto-types

* I-D.ietf-netconf-keystore

* I-D.ietf-netconf-trust-anchors

* I-D.ietf-netmod-factory-default

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

This draft extends the "get-bootstrapping-data" RPC defined in [RFC8572] to include an optional certificate signing request (CSR) [RFC2986], enabling a bootstrapping device to additionally obtain an identity certificate (e.g., an LDevID [Std-802.1AR-2018]) as part of the "onboarding information" response provided in the RPC-reply.

1.2. Terminology

This document uses the following terms from [RFC8572]:

* Bootstrap Server
* Bootstrapping Data
* Conveyed Information
* Device
* Manufacturer
* Onboarding Information
* Signed Data

This document defines the following new terms:

SZTP-client The term "SZTP-client" refers to a "device" that is using a "bootstrap server" as a source of "bootstrapping data".

SZTP-server The term "SZTP-server" is an alternative term for "bootstrap server" that is symmetric with the "SZTP-client" term.

1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. The "ietf-sztp-csr" Module

This section defines a YANG 1.1 [RFC7950] module that augments the "ietf-sztp-bootstrap-server" module defined in [RFC8572] and defines a YANG "structure".

The augmentation adds two nodes ("csr-support" and "csr") to the "input" parameter of the "get-bootstrapping-data" RPC defined in [RFC8572].

The YANG structure, "request-info", defines data returned in the "error-info" node defined in Section 8 of [RFC8572].

2.1. Data Model Overview

The following tree diagram [RFC8340] illustrates the "ietf-sztp-csr" module. The diagram shows the definition of an augmentation adding descendent nodes "csr-support" and "csr" and the definition of a structure called "request-info".

In the order of their intended use:

* The "csr-support" node is used by the SZTP-client to signal to the SZTP-server that it supports the ability the generate CSRs, per this specification. The "csr-support" parameter carries details regarding the SZTP-client's ability to generate CSRs.

* The "request-info" structure is used by the SZTP-server to signal back to the SZTP-client its desire to sign a CSR. The "request-info" structure additionally communicates details about the CSR the SZTP-client is to generate.

* The "csr" node is used by the SZTP-client to communicate its CSR to the SZTP-server. Not shown is how the SZTP-server communicates the signed certificate to the SZTP-client; how to do this is discussed later in this document.

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ==========

module: ietf-sztp-csr

  augment /ietf-sztp-bootstrap-server:get-bootstrapping-data/ietf-sz\
tp-bootstrap-server:input:
    +---- csr-support!
    |  +---- key-generation!
    |  |  +---- supported-algorithms
    |  |     +---- algorithm-identifier*   binary
    |  +---- csr-generation
    |     +---- supported-formats
    |        +---- format-identifier*   identityref
    +---- csr!
       +---- (request-type)
          +--:(p10)
          |  +---- p10?   ietf-crypto-types:csr
          +--:(cmc)
          |  +---- cmc?   binary
          +--:(cmp)
             +---- cmp?   binary

  structure: request-info
     +-- key-generation!
     |  +-- selected-algorithm
     |     +-- algorithm-identifier    binary
     +-- csr-generation
     |  +-- selected-format
     |     +-- format-identifier    identityref
     +-- cert-req-info?    binary
```

To further illustrate how the augmentation and structure defined by the "ietf-sztp-csr" module are used, below are two additional tree diagrams showing these nodes placed where they are used.

The following tree diagram [RFC8340] illustrates SZTP's "get-bootstrapping-data" RPC with the augmentation in place.

```
module: ietf-sztp-bootstrap-server

  rpcs:
    +---x get-bootstrapping-data
       +---w input
       |  +---w signed-data-preferred?   empty
       |  +---w hw-model?                string
       |  +---w os-name?                 string
       |  +---w os-version?              string
       |  +---w nonce?                   binary
       |  +---w sztp-csr:csr-support!
       |  |  +---w sztp-csr:key-generation!
       |  |  |  +---w sztp-csr:supported-algorithms
       |  |  |     +---w sztp-csr:algorithm-identifier*   binary
       |  |  +---w sztp-csr:csr-generation
       |  |     +---w sztp-csr:supported-formats
       |  |        +---w sztp-csr:format-identifier*   identityref
       |  +---w sztp-csr:csr!
       |     +---w (sztp-csr:request-type)
       |        +--:(sztp-csr:p10)
       |        |  +---w sztp-csr:p10?   ct:csr
       |        +--:(sztp-csr:cmc)
       |        |  +---w sztp-csr:cmc?   binary
       |        +--:(sztp-csr:cmp)
       |           +---w sztp-csr:cmp?   binary
       +--ro output
          +--ro reporting-level?    enumeration {onboarding-server}?
          +--ro conveyed-information    cms
          +--ro owner-certificate?      cms
          +--ro ownership-voucher?      cms
```

The following tree diagram [RFC8340] illustrates RESTCONF's "errors" RPC-reply message with the "request-info" structure in place.

```
module: ietf-restconf
  +--ro errors
     +--ro error* []
        +--ro error-type       enumeration
        +--ro error-tag        string
        +--ro error-app-tag?   string
        +--ro error-path?      instance-identifier
        +--ro error-message?   string
        +--ro error-info
           +--ro request-info
              +--ro key-generation!
              |  +--ro selected-algorithm
              |     +--ro algorithm-identifier    binary
              +--ro csr-generation
              |  +--ro selected-format
              |     +--ro format-identifier    identityref
              +--ro cert-req-info?    binary
```

2.2. Example Usage

> The examples below are encoded using JSON, but they could equally well be encoded using XML, as is supported by SZTP.

An SZTP-client implementing this specification would signal to the bootstrap server its willingness to generate a CSR by including the "csr-support" node in its "get-bootstrapping-data" RPC, as illustrated below.

REQUEST

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrappi\
ng-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+json

{
  "ietf-sztp-bootstrap-server:input" : {
    "hw-model": "model-x",
    "os-name": "vendor-os",
    "os-version": "17.3R2.1",
    "nonce": "extralongbase64encodedvalue=",
    "ietf-sztp-csr:csr-support": {
      "key-generation": {
        "supported-algorithms": {
          "algorithm-identifier": [
            "base64encodedvalue1=",
            "base64encodedvalue2=",
            "base64encodedvalue3="
          ]
        }
      },
      "csr-generation": {
        "supported-formats": {
          "format-identifier": [
            "ietf-sztp-csr:p10",
            "ietf-sztp-csr:cmc",
            "ietf-sztp-csr:cmp"
          ]
        }
      }
    }
  }
}
```

Assuming the SZTP-server wishes to prompt the SZTP-client to provide a CSR, then it would respond with an HTTP 400 (Bad Request) error code:

```
RESPONSE
```

```
HTTP/1.1 400 Bad Request
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+json

{
  "ietf-restconf:errors" : {
    "error" : [
      {
        "error-type": "application",
        "error-tag": "missing-attribute",
        "error-message": "Missing input parameter",
        "error-info": {
          "ietf-sztp-csr:request-info": {
            "key-generation": {
              "selected-algortithm": {
                "algorithm-identifier": "base64EncodedValue=="
              }
            },
            "csr-generation": {
              "selected-format": {
                "format-identifier": "ietf-sztp-csr:cmc"
              }
            },
            "cert-req-info": "base64EncodedValue=="
          }
        }
      }
    ]
  }
}
```

Upon being prompted to provide a CSR, the SZTP-client would POST another "get-bootstrapping-data" request, but this time including the "csr" node to convey its CSR to the SZTP-server:

REQUEST

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ==========

POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrappi\
ng-data HTTP/1.1
HOST: example.com
Content-Type: application/yang.data+json

{
  "ietf-sztp-bootstrap-server:input" : {
    "hw-model": "model-x",
    "os-name": "vendor-os",
    "os-version": "17.3R2.1",
    "nonce": "extralongbase64encodedvalue=",
    "ietf-sztp-csr:csr": {
      "p10": "base64encodedvalue=="
    }
  }
}
```

The SZTP-server responds with "onboarding-information" (conveyed encoded inside the "conveyed-information" node) containing a signed identity certificate for the CSR provided by the SZTP-client:

```
RESPONSE

HTTP/1.1 200 OK
Date: Sat, 31 Oct 2015 17:02:40 GMT
Server: example-server
Content-Type: application/yang.data+json

{
  "ietf-sztp-bootstrap-server:output" : {
    "reporting-level": "verbose",
    "conveyed-information": "base64encodedvalue=="
  }
}
```

How the signed certificate is conveyed inside the onboarding information is outside the scope of this document. Some implementations may choose to convey it inside a script (e.g., SZTP's "pre-configuration-script"), while other implementations convey it inside the SZTP "configuration" node.

Following are two examples of conveying the signed certificate inside the "configuration" node. Both examples assume that the SZTP-client understands the "ietf-keystore" module defined in [I-D.ietf-netconf-keystore].

This first example illustrates the case where the signed certificate is for the same asymmetric key used by the SZTP-client's manufacturer-generated identity certificate (e.g., an IDevID). As such, the configuration needs to associate the newly signed certificate with the existing asymmetric key:

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ==========

{
  "ietf-keystore:keystore": {
    "asymmetric-keys": {
      "asymmetric-key": [
        {
          "name": "Manufacturer-Generated Hidden Key",
          "public-key-format": "ietf-crypto-types:subject-public-key\
-info-format",
          "public-key": "base64encodedvalue==",
          "hidden-private-key": [null],
          "certificates": {
            "certificate": [
              {
                "name": "Manufacturer-Generated IDevID Cert",
                "cert": "base64encodedvalue=="
              },
              {
                "name": "Newly-Generated LDevID Cert",
                "cert": "base64encodedvalue=="
              }
            ]
          }
        }
      ]
    }
  }
}
```

This second example illustrates the case where the signed certificate is for a newly generated asymmetric key. As such, the configuration needs to associate the newly signed certificate with the newly generated asymmetric key:

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ==========

{
  "ietf-keystore:keystore": {
    "asymmetric-keys": {
      "asymmetric-key": [
        {
          "name": "Manufacturer-Generated Hidden Key",
          "public-key-format": "ietf-crypto-types:subject-public-key\
-info-format",
          "public-key": "base64encodedvalue==",
          "hidden-private-key": [null],
          "certificates": {
            "certificate": [
              {
                "name": "Manufacturer-Generated IDevID Cert",
                "cert": "base64encodedvalue=="
              }
            ]
          }
        },
        {
          "name": "Newly-Generated Hidden Key",
          "public-key-format": "ietf-crypto-types:subject-public-key\
-info-format",
          "public-key": "base64encodedvalue==",
          "hidden-private-key": [null],
          "certificates": {
            "certificate": [
              {
                "name": "Newly-Generated LDevID Cert",
                "cert": "base64encodedvalue=="
              }
            ]
          }
        }
      ]
    }
  }
}
```

In addition to configuring the signed certificate, it is often necessary to also configure the Issuer's signing certificate so that the the device (i.e., STZP-client) can authenticate certificates presented by peer devices signed by the same issuer as its own. While outside the scope of this document, one way to do this would be to use the "ietf-truststore" module defined in [I-D.ietf-netconf-trust-anchors].

2.3. YANG Module

This module augments an RPC defined in [RFC8572], uses a data type defined in [I-D.ietf-netconf-crypto-types], has an normative references to [RFC2986] and [ITU.X690.2015], and an informative reference to [Std-802.1AR-2018].

<CODE BEGINS> file "ietf-sztp-csr@2020-06-10.yang"

```
module ietf-sztp-csr {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-csr";
  prefix sztp-csr;

  import ietf-sztp-bootstrap-server {
    prefix sztp-svr;
    reference "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
  }

  import ietf-yang-structure-ext {
    prefix sx;
    reference "RFC BBBB:YANG Data Structure Extensions";
  }

  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://tools.ietf.org/wg/netconf
     WG List:  <mailto:netconf@ietf.org>
     Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
               Russ Housley <mailto:housley@vigilsec.com>
               Sean Turner <mailto:sean@sn3rd.com>";

  description
   "This module augments the 'get-bootstrapping-data' RPC,
    defined in the 'ietf-sztp-bootstrap-server' module from
    SZTP (RFC 8572), enabling the SZTP-client to obtain a
    signed identity certificate (e.g., an LDevID from IEEE
    802.1AR) as part of the SZTP 'onboarding-information'
    response.

    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 XXXX
      (https://www.rfc-editor.org/info/rfcXXXX); 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 2020-06-10 {
      description
        "Initial version";
      reference
        "RFC XXXX: Conveying a Certificate Signing Request (CSR)
                   in a Secure Zero Touch Provisioning (SZTP)
                   Bootstrapping Request";
    }

    identity certificate-request-format {
      description
        "A base identity for the request formats supported
         by the SZTP-client.

         Additional derived identities MAY be defined by
         future efforts.";
    }

    identity p10 {
      base "certificate-request-format";
      description
        "Indicates that the SZTP-client supports generating
         requests using the 'CertificationRequest' structure
         defined in RFC 2986.";
      reference
        "RFC 2986: PKCS #10: Certification Request Syntax
                   Specification Version 1.7";
    }
```

```
identity cmc {
  base "certificate-request-format";
  description
    "Indicates that the SZTP-client supports generating
     requests using a constrained version of the 'Full
     PKI Request' structure defined in RFC 5272.";
  reference
    "RFC 5272: Certificate Management over CMS (CMC)";
}

identity cmp {
  base "certificate-request-format";
  description
    "Indicates that the SZTP-client supports generating
     requests that contain a PKCS#10 Certificate Signing
     Request (p10cr), as defined in RFC 2986, encapsulated
     in a Nested Message Content (nested), as defined in
     RFC 4210.";
  reference
    "RFC 2986: PKCS #10: Certification Request Syntax
               Specification Version 1.7
     RFC 4210: Internet X.509 Public Key Infrastructure
               Certificate Management Protocol (CMP)";
}


// Protocol-accessible nodes

augment "/sztp-svr:get-bootstrapping-data/sztp-svr:input" {

  description
    "This augmentation adds the 'csr-support' and 'csr' nodes to
     the SZTP (RFC 8572) 'get-bootstrapping-data' request message,
     enabling the SZTP-client to obtain an identity certificate
     (e.g., an LDevID from IEEE 802.1AR) as part of the onboarding
     information response provided by the SZTP-server.

     The 'csr-support' node enables the SZTP-client to indicate
     that it supports generating certificate signing requests
     (CSRs), and to provide details around the CSRs it is able
     to generate.

     The 'csr' node enables the SZTP-client to relay a CSR to
     the SZTP-server.";

   reference
     "IEEE 802.1AR: IEEE Standard for Local and metropolitan
                    area networks - Secure Device Identity
```

```
          RFC 8572: Secure Zero Touch Provisioning (SZTP)";

      container csr-support {
        presence
          "Indicates that the SZTP-client is capable of sending CSRs.";
        description
          "The 'csr-support' node enables the SZTP-client to indicate
           that it supports generating certificate signing requests
           (CSRs), and to provide details around the CSRs it is able
           to generate.

           When present, the SZTP-server MAY respond with the HTTP
           error 400 (Bad Request) with an 'ietf-restconf:errors'
           document having the 'error-tag' value 'missing-attribute'
           and the 'error-info' node containing the 'request-info'
           structure described in this module.";
        container key-generation {
          presence
            "Indicates that the SZTP-client is capable of
             generating a new asymmetric key pair.

             If this node is not present, the SZTP-server MAY
             request a CSR using the asymmetric key associated
             with the device's existing identity certificate
             (e.g., an LDevID from IEEE 802.1AR).";
          description
            "Specifies details for the SZTP-client's ability to
             generate a new asymmetric key pair.";
          container supported-algorithms {
            description
              "A list of public key algorithms supported by the
               SZTP-client for generating a new key.";
            leaf-list algorithm-identifier {
              type binary;
              min-elements 1;
              description
                "An AlgorithmIdentifier, as defined in RFC 2986,
                 encoded using ASN.1 distinguished encoding rules
                 (DER), as specified in ITU-T X.690.";
              reference
                "RFC 2986: PKCS #10: Certification Request Syntax
                           Specification Version 1.7
                 ITU-T X.690:
                   Information technology - ASN.1 encoding rules:
                   Specification of Basic Encoding Rules (BER),
                   Canonical Encoding Rules (CER) and Distinguished
                   Encoding Rules (DER).";
            }
```

```
          }
        }
        container csr-generation {
          description
            "Specifies details for the SZTP-client's ability to
             generate a certificate signing requests.";
          container supported-formats {
            description
              "A list of certificate request formats supported
               by the SZTP-client for generating a new key.";
            leaf-list format-identifier {
              type identityref {
                base certificate-request-format;
              }
              min-elements 1;
              description
                "A certificate request format supported by the
                 SZTP-client.";
            }
          }
        }
      }

      container csr {
        presence
          "Indicates that the SZTP-client has sent a CSR.";
        description
          "The 'csr' node enables the SZTP-client to convey
           a certificate signing request, using the encoding
           format selected by the SZT-server's 'request-info'
           response to the SZTP-client's previously sent
           'get-bootstrapping-data' request containing the
           'csr-support' node.

           When present, the SZTP-server SHOULD respond with
           an SZTP 'onboarding-information' message containing
           a signed certificate for the conveyed CSR.  The
           SZTP-server MAY alternatively respond with another
           HTTP error containing another 'request-info', in
           which case the SZTP-client MUST invalidate the CSR
           sent in this node.";
        choice request-type {
          mandatory true;
          description
            "A choice amongst certificate signing request formats.

             Additional formats MAY be augmented into this 'choice'
             statement by future efforts.";
```

```
          case p10 {
            leaf p10 {
              type ct:csr;
              description
                "A CertificationRequest structure, per RFC 2986.
                 Please see 'csr' in RFC AAAA for encoding details.";
              reference
                "RFC 2986:
                   PKCS #10: Certification Request Syntax Specification
                 RFC AAAA:
                   Common YANG Data Types for Cryptography";
            }
          }
          case cmc {
            leaf cmc {
              type binary;
              description
                "A constrained version of the 'Full PKI Request'
                 message defined in RFC 5272, encoded using ASN.1
                 distinguished encoding rules (DER), as specified
                 in ITU-T X.690.

                 For asymmetric key-based origin authentication
                 of a CSR based on the IDevID's private key for the
                 associated IDevID's public key, the PKIData contains
                 one reqSequence element and no controlSequence,
                 cmsSequence, or otherMsgSequence elements. The
                 reqSequence is the TaggedRequest and it is the tcr
                 CHOICE. The tcr is the TaggedCertificationRequest
                 and it a bodyPartId and the certificateRequest
                 elements. The certificateRequest is signed with
                 the IDevID's private key.

                 For asymmetric key-based origin authentication
                 based on the IDevID's private key that encapsulates
                 a CSR signed by the LDevID's private key, the
                 PKIData contains one cmsSequence element and no
                 controlSequence, reqSequence, or otherMsgSequence
                 elements. The cmsSequence is the TaggedContentInfo
                 and it includes a bodyPartID element and a
                 contentInfo. The contentInfo is a SignedData
                 encapsulating a PKIData with one reqSequence
                 element and no controlSequence, cmsSequence, or
                 otherMsgSequence elements. The reqSequence is
                 the TaggedRequest and it is the tcr CHOICE. The
                 tcr is the TaggedCertificationRequest and it a
                 bodyPartId and the certificateRequest elements.
                 The certificateRequest is signed with the LDevID's
```

```
                private key.

                For shared secret-based origin authentication of
                a CSR signed by the LDevID's private key, the
                PKIData contains one cmsSequence element and no
                controlSequence, reqSequence, or otherMsgSequence
                elements. The cmsSequence is the TaggedContentInfo
                and it includes a bodyPartID element and a
                contentInfo. The contentInfo is an AuthenticatedData
                encapsulating a PKIData with one reqSequence
                element and no controlSequence, cmsSequence, or
                otherMsgSequence elements. The reqSequence is the
                TaggedRequest and it is the tcr CHOICE. The tcr
                is the TaggedCertificationRequest and it a
                bodyPartId and the certificateRequest elements.
                The certificateRequest is signed with the LDevID's
                private key.";
              reference
                "RFC 5272: Certificate Management over CMS (CMC)
                 ITU-T X.690:
                   Information technology - ASN.1 encoding rules:
                   Specification of Basic Encoding Rules (BER),
                   Canonical Encoding Rules (CER) and Distinguished
                   Encoding Rules (DER).";
            }
          }
          case cmp {
            leaf cmp {
              type binary;
              description
                "A PKIMessage structure, as defined in RFC 4210,
                 encoded using ASN.1 distinguished encoding rules
                 (DER), as specified in ITU-T X.690.

                 The PKIMessage structure contains a PKCS#10
                 Certificate Signing Request (p10cr), as defined in
                 RFC 2986, encapsulated in a Nested Message Content
                 (nested) structure, as defined in RFC 4210.;

                 For asymmetric key-based origin authentication of
                 a CSR based on the IDevID's private key for the
                 associated IDevID's public key, PKIMessages contains
                 one PKIMessage with one body element, a header
                 element that is an empty sequence, and no protection
                 or extraCerts elements. The body element contains a
                 p10cr CHOICE.

                 For asymmetric key-based origin authentication based
```

```
                on the IDevID's private key that encapsulates a CSR
                signed by the LDevID's private key, PKIMessages
                contains one PKIMessage with one header element,
                one body element, one protection element, and one
                extraCerts element.  The header element contains
                pvno, sender, recipient, and protectionAlg elements
                and no other elements. The body element contains the
                nested CHOICE. The nested element's PKIMessages
                contains one PKIMessage with one body element, one
                header element that is an empty sequence, and no
                protection or extraCerts elements. The nested
                element's body element contains a p10cr CHOICE. The
                protection element contains the digital signature
                generated with the IDevID's private key. The
                extraCerts element contains the IDevID certificate.

                For shared secret-based origin authentication of a
                CSR signed by the LDevID's private key, PKIMessages
                contains one PKIMessage with one header element,
                one body element, one protection element, and no
                extraCerts element. The header element contains
                pvno, sender, recipient, and protectionAlg elements
                and no other elements. The body element contains
                the nested CHOICE. The nested element's PKIMessages
                contains one PKIMessage with one body element, one
                header element that is an empty sequence, and no
                protection or extraCerts elements. The body element
                contains a p10cr CHOICE. The protection element
                contains the MAC value generated with the shared
                secret.";
            reference
              "RFC 2986:
                 PKCS #10: Certification Request Syntax
                 Specification Version 1.7
               RFC 4210:
                 Internet X.509 Public Key Infrastructure
                 Certificate Management Protocol (CMP)
               ITU-T X.690:
                 Information technology - ASN.1 encoding rules:
                 Specification of Basic Encoding Rules (BER),
                 Canonical Encoding Rules (CER) and Distinguished
                 Encoding Rules (DER).";
          }
        }
      }
    }
  }
```

```
    sx:structure request-info {
      container key-generation {
        presence
          "Indicates that the SZTP-client is to generate a new
           asymmetric key.  If missing, then the SZTP-client
           MUST reuse the key associated with its existing
           identity certificate (e.g., IDevID).

           This leaf MUST only appear if the SZTP-clients
           'csr-support' included the 'key-generation' node.";
        description
          "Specifies details for the key that the SZTP-client
           is to generate.";
        container selected-algorithm {
          description
            "The key algorithm selected by the SZTP-server. The
             algorithm MUST be one of the algorithms specified
             by the 'supported-algorithms' node in the
             SZTP-client's request message.";
          leaf algorithm-identifier {
            type binary;
            mandatory true;
            description
              "An AlgorithmIdentifier, as defined in RFC 2986,
               encoded using ASN.1 distinguished encoding rules
               (DER), as specified in ITU-T X.690.";
            reference
              "RFC 2986: PKCS #10: Certification Request Syntax
                         Specification Version 1.7
               ITU-T X.690:
                 Information technology - ASN.1 encoding rules:
                 Specification of Basic Encoding Rules (BER),
                 Canonical Encoding Rules (CER) and Distinguished
                 Encoding Rules (DER).";
          }
        }
      }
      container csr-generation {
        description
          "Specifies details for the CSR that the SZTP-client
           is to generate.";
        container selected-format {
          description
            "The CSR format selected by the SZTP-server. The
             format MUST be one of the formats specified by
             the 'supported-formats' node in the SZTP-client's
             request message.";
          leaf format-identifier {
```

```
            type identityref {
              base certificate-request-format;
            }
            mandatory true;
            description
              "A certificate request format to be used by the
               SZTP-client.";
          }
        }
      }
      leaf cert-req-info {
        type binary;
        description
          "A CertificationRequestInfo structure, as defined in
           RFC 2986, encoded using ASN.1 distinguished encoding
           rules (DER), as specified in ITU-T X.690.

           Enables the SZTP-server to provide a fully-populated
           CertificationRequestInfo structure that the SZTP-client
           only needs to sign in order to generate the complete
           'CertificationRequest' structure to send to SZTP-server
           in its next 'get-bootstrapping-data' request message.

           When provided, the SZTP-client SHOULD use this
           structure to generate its CSR; failure to do so MAY
           result in another 400 (Bad Request) response.

           When not provided, the SZTP-client SHOULD generate a
           CSR using the same structure defined in its existing
           identity certificate (e.g., IDevID).

           It is an error if the 'AlgorithmIdentifier' field
           contained inside the 'SubjectPublicKeyInfo' field
           does not match the algorithm identified by the
           'selected-algorithm' node.";
        reference
          "RFC 2986: PKCS #10: Certification Request Syntax
                     Specification Version 1.7
           ITU-T X.690:
             Information technology - ASN.1 encoding rules:
             Specification of Basic Encoding Rules (BER),
             Canonical Encoding Rules (CER) and Distinguished
             Encoding Rules (DER).";
      }
    }
  }

  <CODE ENDS>
```

3. Security Considerations

This document builds on top of the solution presented in [RFC8572] and therefore all the Security Considerations discussed in RFC 8572 apply here as well.

3.1. SZTP-Client Considerations

3.1.1. Ensuring the Integrity of Asymmetric Private Keys

The private key the SZTP-client uses for the dynamically-generated identity certificate MUST be protected from inadvertent disclosure in order to prevent identity fraud.

The security of this private key is essential in order to ensure the associated identity certificate can be used as a root of trust.

It is RECOMMENDED that devices are manufactured with an HSM (hardware security module), such as a TPM (trusted platform module), to generate and forever contain the private key within the security perimeter of the HSM. In such cases, the private key, and its associated certificates, MAY have long validity periods.

In cases where the device does not possess an HSM, or otherwise is unable to use an HSM for the private key, it is RECOMMENDED to regenerate the private key (and associated identity certificates) periodically. Details for how to generate a new private key and associate a new identity certificate are outside the scope of this document.

3.1.2. Reuse of a Manufacturer-generated Private Key

It is RECOMMENDED in [RFC8572] that devices are shipped from manufacturing with a secure device identity certificate (e.g., an IDevID, from [Std-802.1AR-2018]). It is also RECOMMENDED that the private key for these necessarily long-lived certificates be stored in an HSM, such as a TPM. Lastly, per the previous consideration, when devices generate a new private key, it is also RECOMMENDED that the private key is protected by the HSM.

However, it is understood that space on an HSM chip may be limited, potentially to the point of not being able to store an additional private key for the CSR described in this document, and that it may not be possible to store hardware-protected keys outside the TPM (e.g., a TPM-encrypted key stored in non-volatile memory). In such cases, it is RECOMMENDED to reuse the existing hardware-protected private key rather than generate a second private key outside of protection afforded by the hardware.

3.1.3. Replay Attack Protection

This RFC enables an SZTP-client to announce an ability to generate new key to use for its CSR.

When the SZTP-server responds with a request for the device to generate a new key, it is essential that the device actually generates a new key.

Generating a new key each time enables the random bytes used to create the key to serve the dual-purpose of also acting like a "nonce" used in other mechanisms to detect replay attacks.

When a fresh public/private key pair is generated for the request, confirmation to the SZTP-client that the response has not been replayed is enabled by the SZTP-client's fresh public key appearing in the signed certificate provided by the SZTP-server.

When a public/private key pair associated with the IDevID used for the request, there may not be confirmation to the SZTP-client that the response has not been replayed; however, the worst case result is a lost certificate that is associated to the private key known only to the SZTP-client.

3.1.4. Connecting to an Untrusted Bootstrap Server

[RFC8572] allows SZTP-clients to connect to untrusted SZTP-servers, by blindly authenticating the SZTP-server's TLS end-entity certificate.

As is discussed in Section 9.5 of [RFC8572], in such cases the SZTP-client MUST assert that the bootstrapping data returned is signed, if the SZTP-client is to trust it.

However, the HTTP error message used in this document cannot be signed data, as described in RFC 8572.

Therefore, the solution presented in this document cannot be used when the SZTP-client connects to an untrusted SZTP-server.

Consistent with the recommendation presented in Section 9.6 of [RFC8572], SZTP-clients SHOULD NOT passed the "csr-support" input parameter to an untrusted SZTP-server. SZTP-clients SHOULD pass instead the "signed-data-preferred" input parameter, as discussed in Appendix B of [RFC8572].

3.1.5. Selecting the Best Origin Authentication Mechanism

When generating a new key, it is important that the client be able to provide additional proof to the CA that it was the entity that generated the key.

All of the certificate request formats defined in this document (e.g., CMS, CMP, etc.), not including a raw PKCS#10, support origin authentication.

These formats support origin authentication using both PKI and shared secret.

Typically only one possible origin authentication mechanism can possibly be used but, in the case that the SZTP-client authenticates itself using both TLS-level (e.g., IDevID) and HTTP-level credentials (e.g., Basic), as is allowed by Section 5.3 of [RFC8572], then the SZTP-client may need to choose between the two options.

In the case the SZTP-client must choose between the asymmetric key option versus a shared secret for origin authentication, it is RECOMMENDED that the SZTP-client choose using the asymmetric key option.

3.1.6. Clearing the Private Key and Associated Certificate

Unlike a manufacturer-generated identity certificate (e.g., IDevID), the deployment-generated identity certificate (e.g., LDevID) and the associated private key (assuming a new private key was generated for the purpose), are considered user data and SHOULD be cleared whenever the device is reset to its factory default state, such as by the "factory-reset" RPC defined in [I-D.ietf-netmod-factory-default].

3.2. SZTP-Server Considerations

3.2.1. Conveying Proof of Possession to a CA

3.2.2. Supporting SZTP-Clients that don’t trust the SZTP-Server

[RFC8572] allows SZTP-clients to connect to untrusted SZTP-servers, by blindly authenticating the SZTP-server’s TLS end-entity certificate.

As is recommended in Section 3.1.4 in this document, in such cases, SZTP-clients SHOULD pass the "signed-data-preferred" input parameter.

The reciprocal of this statement is that SZTP-servers, wanting to support SZTP-clients that don't trust them, SHOULD support the "signed-data-preferred" input parameter, as discussed in Appendix B of [RFC8572].

3.2.3. YANG Module Considerations

The recommended format for documenting the Security Considerations for YANG modules is described in Section 3.7 of [RFC8407]. However, the module defined in this document only augments two input parameters into the "get-bootstrapping-data" RPC in [RFC8572], and therefore only needs to point to the relevant Security Considerations sections in that RFC.

* Security considerations for the "get-bootstrapping-data" RPC are described in Section 9.16 of [RFC8572].

* Security considerations for the "input" parameters passed inside the "get-bootstrapping-data" RPC are described in Section 9.6 of [RFC8572].

4. IANA Considerations

4.1. The IETF XML Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688] maintained at https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-sztp-csr
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

4.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020] maintained at https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml. Following the format defined in [RFC6020], the below registration is requested:

name: ietf-sztp-csr
namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-csr
prefix: sztp-csr
reference: RFC XXXX

5. References

5.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[ITU.X690.2015]
International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1, August 2015, <https://www.itu.int/rec/T-REC-X.690/>.

[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>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[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>.

[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>.

[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>.

5.2. Informative References

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-17, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-17>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

[I-D.ietf-netmod-factory-default]
WU, Q., Lengyel, B., and Y. Niu, "A YANG Data Model for Factory Default Settings", Work in Progress, Internet-Draft, draft-ietf-netmod-factory-default-15, 25 April 2020, <https://tools.ietf.org/html/draft-ietf-netmod-factory-default-15>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[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>.

[Std-802.1AR-2018]
Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", 14 June 2018, <http://standards.ieee.org/findstds/standard/802.1AR-2018.html>.

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Russ Housley
Vigil Security, LLC

Email: housley@vigilsec.com

Sean Turner
sn3rd

Email: sean@sn3rd.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2021

Q. Wu
Huawei
L. Geng
P. Liu
China Mobile

July 12, 2020

Telemetry Data Export capability
draft-tao-netconf-data-export-capabilities-01

Abstract

This document proposes a YANG module for telemetry data export capability which augments system Capabilities model and provides additional telemetry data export attributes associated with system capability for transport dependent capability negotiation.

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 January 13, 2021.

Copyright Notice

Table of Contents

1. Introduction

Notification capability model defined in [I-D.netconf-notification-capabilities] allows a client to discover a set of capabilities supported by the server (e.g., basic system capability and YANG-Push related capabilities) both at implementation-time and run-time. These "capabilities" permit the client to adjust its behavior to take advantage of the features exposed by the device.

However pre-configuration for some transport specific parameters (e.g., transport protocol, encoding format, encryption by the client is still inevitable, which may cause unexpected failure and additional message exchange between client and server.

This document proposes a YANG module for telemetry data export capability which augments System Capabilities model and provide additional data export attributes for transport dependent capability negotiation.

1.1. Terminology

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.

2. Data Export capability

The YANG module ietf-notification-capabilities defined in [I-D.netconf-notification-capabilities] specify the following server capabilities related to YANG Push:

- A set of capabilities related to the amount of notifications the server can send out
- Specification of which data nodes support on-change notifications.
- Capability values can be specified on server level, datastore level or on specific data nodes (and their contained sub-tree) of a specific datastore. Capability values on a smaller, more specific part of the server's data always override more generic values.
- On-change capability is not specified on a server level as different datastores usually have different on-change capabilities. On a datastore level on-change capability for configuration and state data can be specified separately.

These server capabilities are transport independent and session level capabilities and can be provided either at implementation time or reported at run time.

This document augments system Capabilities model and provides additional data export attributes associated with system capabilities:

- Specification of transport protocol the client can use to establish transport connection;
- Specification of encoding selection(e.g., XML or JSON, to binary) of Data Modeled with YANG;
- Specification of secure transport mechanisms that are needed by the client to communicate with the server;
- Specification of the type of data compression algorithm (e.g., lossless data compression) the client can use for file compression and decompression
- Specification of Maximum number of data nodes that can be sent in a group of data node with the same characteristics;

- Specification of the number of sensors group. A sensor group represents a resuable grouping of multiple paths and exclude filters.

- Specification of the notification message encapsulation type, either one notification per message or multiple notifications per message.

- Specification of the type of subscription, e.g., periodic subscrption, on-change subscription, bulk subscription, adaptive subscription.

- Specification of the update trigger type such as timer event based trigger, count threshold trigger, redundant suppression.

2.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```
module: ietf-data-export-capabilities
  augment /sysc:system-capabilities:
    +--ro data-export-capabilities
       +--ro transport-protocol?            identityref
       +--ro encoding-format?               identityref
       +--ro security-protocol?             identityref
       +--ro compression-mode?              identityref
       +--ro max-nodes-per-sensor-group?    uint32
       +--ro max-sensor-group-per-update?   uint32
  augment /sysc:system-capabilities/inc:subscription-capabilities:
    +--ro data-export-capabilities
       +--ro message-bundling-support?   boolean
       +--ro subscription-mode?          identityref
  augment /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-cap
abilities:
    +--ro data-export-capabilities
       +--ro timer-event-support?         boolean
       +--ro sampling-interval* []
       |  +--ro observable-period         centiseconds
       |  +--ro count?         uint16
       |  +--ro anchor-time?   yang:date-and-time
       +--ro counter-threshold-support?   boolean
       +--ro suppress-redundant?          boolean
```

3. YANG Module

```
<CODE BEGINS> file "ietf-data-export-capabilities.yang"
module ietf-data-export-capabilities {
  yang-version 1.1;
```

```
  namespace "urn:ietf:params:xml:ns:yang:ietf-data-export-capabilities";
  prefix dec;

  import ietf-system-capabilities {
    prefix sysc;
  }
  import ietf-notification-capabilities {
    prefix inc;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <https://tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>
     Editor:   Qin Wu
               <mailto:bill.wu@huawei.com";
  description
    "This module defines an extension to System Capability and YANG Push
     Notification Capabilities model and provides additional data export
     attributes for transport dependent capability negotiation.

     Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the Simplified BSD License
     set forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.";

  revision 2020-07-03 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: Telemetry Data Export capability";
  }

  identity transport-protocol {
    description
      "Base identity for transport protocol type.";
```

```
  }

  identity tcp {
    base transport-protocol;
    description
      "Identity for tcp transport protocol.";
  }

  identity udp {
    base transport-protocol;
    description
      "Identity for udp transport protocol.";
  }

  identity grpc {
    base transport-protocol;
    description
      "Identity for grpc transport protocol.";
  }

  identity security-protocol {
    description
      "Base identity for security protocol type.";
  }

  identity tls {
    base security-protocol;
    description
      "Identity for tls security protocol.";
  }

  identity ssh {
    base security-protocol;
    description
      "Identity for ssh transport protocol.";
  }

  identity encoding-format {
    description
      "Base identity for encoding format type.";
  }

  identity xml {
    base encoding-format;
    description
      "Identity for xml encoding format.";
  }
```

```
  identity json {
    base encoding-format;
    description
      "Identity for json encoding format.";
  }

  identity gpb {
    base encoding-format;
    description
      "Identity for gpb encoding format.";
  }

  identity cbor {
    base encoding-format;
    description
      "Identity for cbor encoding format.";
  }

  identity compression-mode {
    description
      "Base identity for compression mode.";
  }

  identity gzip {
    base security-protocol;
    description
      "Identity for gzip compression mode.";
  }

  identity deflate {
    base security-protocol;
    description
      "Identity for deflate compression mode.";
  }

  identity subscription-mode {
    description
      "Base identity for subscription mode.";
  }

  identity periodic {
    base subscription-mode;
    description
      "Identity for periodic subscription mode.";
  }

  identity on-change {
    base subscription-mode;
```

```
    description
      "Identity for on change subscription mode.";
  }

  identity event {
    base subscription-mode;
    description
      "Identity for event based subscription mode.";
  }

  typedef centiseconds {
    type uint32;
    description
      "A period of time, measured in units of 0.01 seconds.";
  }

  augment "/sysc:system-capabilities" {
    description
      "Add system level capability.";
    container data-export-capabilities {
      description
        "Capabilities related to telemetry data export capability negotiation.";
      leaf transport-protocol {
        type identityref {
          base transport-protocol;
        }
        description
          "Type of transport protocol.";
      }
      leaf encoding-format {
        type identityref {
          base encoding-format;
        }
        description
          "Type of encoding format.";
      }
      leaf security-protocol {
        type identityref {
          base security-protocol;
        }
        description
          "Type of secure transport.";
      }
      leaf compression-mode {
        type identityref {
          base compression-mode;
        }
        description
```

```
          "Type of compression mode.";
      }
      leaf max-nodes-per-sensor-group {
        type uint32 {
          range "1..max";
        }
        description
          "Maximum number of selected data nodes that can be sent
           per sensor group.";
      }
      leaf max-sensor-group-per-update {
        type uint32 {
          range "1..max";
        }
        description
          "Maximum number of sensor groups that can be sent
           in an update.";
      }
    }
  }
  augment "/sysc:system-capabilities/inc:subscription-capabilities" {
    description
      "Add subscription level capability.";
    container data-export-capabilities {
      description
        "Capabilities related to telemetry data export capability negotiation.";
      leaf message-bundling-support {
        type boolean;
        default "false";
        description
          "Enables message bundling support.";
      }
      leaf subscription-mode {
        type identityref {
          base subscription-mode;
        }
        description
          "Type of subscription mode.";
      }
    }
  }
  augment "/sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-ca
pabilities" {
    description
      "Add datastore and node level capability.";
    container data-export-capabilities {
      description
        "Capabilities related to telemetry data export capability negotiation.";
      leaf timer-event-support {
```

```
        type boolean;
        default "false";
        description
          "Set to true if the subscription mode is event based
           subscription mode and timer based trigger is supported.
           Set to false if event based subscription mode is not
           supported.";
      }
      list sampling-interval {
        description
          "Time-based triggers are used to define the
           Sampling intervals.  All packets are selected that arrive
           at the Observation Point within the time intervals defined
           by the start and stop triggers (i.e., arrival time of the
           packet is larger than the start time and smaller than the
           stop time).";
        leaf observable-period {
          type centiseconds;
          mandatory true;
          description
            "Duration of time that should occur between Observation Point
             for periodic push updates, in units of 0.01 seconds.";
        }
        leaf count {
          type uint16;
          description
            "specify the count number of interval that has to pass before
             successive adaptive periodic push update records for the same
             subscription are generated for a receiver.";
        }
        leaf anchor-time {
          type yang:date-and-time;
          description
            "Designates a timestamp before or after which a series
             of periodic push updates are determined.  The next
             update will take place at a point in time that is a
             multiple of a period from the 'anchor-time'.
             For example, for an 'anchor-time' that is set for the
             top of a particular minute and a period interval of a
             minute, updates will be sent at the top of every
             minute that this subscription is active.";
        }
      }
      leaf counter-threshold-support {
        type boolean;
        default "false";
        description
          "Set to true if the subscription mode is event based
```

```
            subscription mode and counter based trigger is support
            (i.e., named counter crosses a specified threshold).
            Set to false if event based subscription mode is not
            supported.";
        }
        leaf suppress-redundant {
          type boolean;
          default "false";
          description
            "Suppress dupplicated data objects to be sent during each update interv
al.";
        }
      }
    }
  }
}
<CODE ENDS>
```

4. IANA Considerations

4.1. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

```
URI:
   urn:ietf:params:xml:ns:yang:ietf-data-export-capabilities
Registrant Contact:
   The IESG.
XML:
   N/A; the requested URI is an XML namespace.
```

4.2. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
name:
   ietf-data-export-capabilities
namespace:
   urn:ietf:params:xml:ns:yang:ietf-data-export-capabilities
prefix:
   dec
reference:
   RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
   this note.)
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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 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:

- /sysc:system-capabilities/dec:transport-protocol
- /sysc:system-capabilities/dec:encoding-format
- /sysc:system-capabilities/dec:secure-transport
- /sysc:system-capabilities/dec:compression-mode
- /sysc:system-capabilities/dec:max-nodes-per-sensor-group
- /sysc:system-capabilities/dec:sensor-group-count
- /sysc:system-capabilities/inc:subscription-capabilities/ dec:message-bundling-support
- /sysc:system-capabilities/inc:subscription-capabilities/ dec:subscription-mode
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/dec:sampling-interval

6. Contributors

The authors would like to thank Ran Tao for his major contributions to the initial modeling and use cases.

7. References

7.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

7.2. 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>.

[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>.

Appendix A. Usage Example of interaction between telemetry data export capabilities and Adaptive Subscription

The following instance-data example describes the notification capabilities of a hypothetical "acme-router". The router implements the running, and operational datastores. Every change can be reported on-change from running, but only config=true nodes and some config=false data from operational. Interface statistics are reported only when both timer-event-support and count-threshold-support are set to true.

```
<CODE BEGINS> file "acme-router-notification-capabilities.xml"
========== NOTE: '\' line wrapping per BCP YYY (RFC YYYY) ===========

<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=\
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
```

```
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Defines the notification capabilities of an acme-router.
    The router only has running, and operational datastores.
    Every change can be reported on-change from running, but
    only config=true nodes and some config=false data from operational.
    Statistics are not reported based on timer based trigger and counter
    threshold based trigger.
  </description>
  <content-data>
    <system-capabilities \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities" \
      xmlns:inc=\
        "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities" \
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics\
          </node-selector>
          <inc:subscription-capabilities>
            <inc:minimum-dampening-period>5
              </inc:minimum-dampening-period>
            <inc:on-change-supported>\
              state-changes\
            </inc:on-change-supported>
          </inc:subscription-capabilities>
        </per-node-capabilities>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics/if:out-octets\
          </node-selector>
          <dec:data-export-capabilities>
            <dec:timer-event-support>true</dec:timer-event-support>
              <dec:sampling-interval>
              <dec:period>5</dec:period>
              <dec:count>6</dec:count>
              </dec:sampling-interval>
              <dec:sampling-interval>
              <dec:period>60</dec:period>
              <dec:count>6</dec:count>
              </dec:sampling-interval>
              <dec:threshold-event-support>false</dec:threshold-event-support>
          </dec:data-export-capabilities>
```

```
        </per-node-capabilities>
        </per-node-capabilities>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics/if:in-errors\
          </node-selector>
          <dec:data-export-capabilities>
            <dec:timer-event-support>false</dec:timer-event-support>
            <dec:threshold-event-support>true</dec:threshold-event-support>
          </dec:data-export-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

The client configure adaptive subscription parameters on the server. The adaptive subscription configuration parameters require the server to scan all interface of specific type every 5 seconds up to 30 seconds if the value of interface in-errors is greater than 1000 ; If the interface in-errors value is less than 1000, switch to 60 seconds period value, and then scan all client every 60 seconds up to 360 seconds. 30 seconds and 360 seconds can be seen as time window. The time window lenght is 6 period values. Irrespective of period value set for adaptive subscription, 6 event records during the time window should be generated for the same subscription and send to the receivers.

```
     <rpc message-id="101"
          xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <edit-config>
         <target>
           <running/>
         </target>
         <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
           <top xmlns="http://example.com/schema/1.2/config">
           <yp:datastore
             xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
             ds:running
           </yp:datastore>
          <yp:datastore-xpath-filter
            xmlns:ex="https://example.com/sample-data/1.0">
            /if:ietf-interfaces
         </yp:datastore-xpath-filter>
       <as:adaptive-subscriptions
      xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
        <as:data-path>/if:interfaces/if:interface/if:statistics</as:data-path>
        <as:target>in-errors</as:target>
       <as:adaptive-period>
        <as:condition-expression>in-errors < 1000</as:condition-expressioni>
        <as:watermark>1000</as:watermark>
        <as:period>5</as:period>
         <as:count>12</as:count>
       </as:adaptive-period>
       <as:adaptive-period>
        <as:condition-expression>in-errors < 1000</as:condition-expressioni>
        <as:watermark>1000</as:watermark>
        <as:period>60</as:period>
         <as:count>12</as:count>
       </as:adaptive-period>
      </as:adaptive-subscriptions>
           </top>
         </config>
       </edit-config>
     </rpc>
```

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 August 2022

Q. Wu
Q. Ma
Huawei
P. Liu
China Mobile
W. Wang
China Telecom

24 February 2022

Data Export Notification Capability
draft-tao-netconf-data-export-capabilities-07

Abstract

This document proposes a YANG module for data export notification capabilities which augments "ietf-system-capabilities" YANG module defined in [RFC9196] and provides additional data export attributes associated with system capabilities for transport specific Notification. This YANG module can be used by the client to learn capability information from the server at runtime or at implementation time, by making use of the YANG instance data file format.

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 28 August 2022.

Copyright Notice

Table of Contents

1. Introduction

Notification capabilities model defined in [RFC9196] allows a client to discover a set of capabilities supported by the server (e.g., basic system capabilities and YANG-Push related capabilities) both at implementation time and at runtime. These capabilities allow the client to adjust its behavior to take advantage of the features exposed by the server.

However the client and the server may still support various different transport specific parameters (e.g., transport protocol, encoding format, encryption). As described in section 3.1 of [RFC8641], a simple negotiation (i.e., inserting hints into error responses to a failed RPC request) between subscribers and publishers for subscription parameters increases the likelihood of success for subsequent RPC requests, but not guaranteed, which may cause unexpected failure or additional message exchange between client and server.

This document defines a corresponding solution that is built on top of [RFC9196]. Supplementing that work are YANG data model augmentations for transport specific notification. The module can be used by the client to discover capability information from the server at runtime or at implementation time, by making use of the YANG instance data file format.

1.1. Terminology

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.

2. Data Export Capability

The YANG module "ietf-notification-capabilities" defined in [RFC9196] specifies the following server capabilities related to YANG Push:

* Supported (reporting) periods for "periodic" subscriptions

* Maximum number of objects that can be sent in an update

* The set of datastores or data nodes for which "periodic" or "on-change" notification is supported

* Supported dampening periods for "on-change" subscriptions

These server capabilities are transport independent, session level capabilities. They can be provided either at the implementation time or reported at runtime.

This document augments System Capabilities model and provides additional data export attributes associated with system capabilities:

* Specification of transport protocols the client can use to establish a transport connection;

* Specification of the encoding selection used (e.g., XML or JSON, Binary) for Data modeled with YANG;

* Specification of secure transport mechanisms that are needed by the client to communicate with the server;

* Specification of the notification message encapsulation type, either one notification per message or multiple notifications per message [I-D. ietf-netconf-notification-messages].

2.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```
module: ietf-data-export-capabilities
  augment /sysc:system-capabilities:
    +--ro data-export-capabilities
       +--ro data-export-capability* []
          +--ro transport-protocol?            identityref
          +--ro encoding-format*               identityref
          +--ro security-protocol?             identityref
          +--ro message-bundling-support?      empty
```

3. YANG Module

```
<CODE BEGINS> file "ietf-data-export-capabilities.yang"
module ietf-data-export-capabilities {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-data-export-capabilities";
  prefix dec;

  import ietf-system-capabilities {
    prefix sysc;
  }
  import ietf-notification-capabilities {
    prefix inc;
  }
  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <https://tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>
     Editor:   Qin Wu <mailto:bill.wu@huawei.com>
     Editor:   Qiufang Ma <mailto: maqiufang1@huawei.com>
     Editor:   Peng Liu <mailto: liupengyjy@chinamobile.com>
     Editor:   Wei Wang <mailto: wangw36@chinatelecom.cn>";
  description
    "This module defines an extension to System Capability and YANG Push
     Notification Capabilities model and provides additional data export
     attributes for transport specific notification.

     Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
```

```
        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX;
        see the RFC itself for full legal notices.";

    revision 2020-07-03 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: Data Export Notification Capability";
    }

    identity transport-protocol {
      description
        "Base identity for transport protocol type.";
    }

    identity tcp {
      base transport-protocol;
      description
        "Identity for tcp as transport protocol.";
    }

    identity udp-notif {
      base transport-protocol;
      description
        "Identity for udp notif as transport protocol.";
      reference
        "draft-ietf-netconf-udp-notif:UDP-based Transport
         for Configured Subscriptions";
    }

    identity http-notif {
      base transport-protocol;
      description
        "Identity for http notif as transport protocol.";
      reference
        "draft-ietf-netconf-https-notif: An HTTPS-based
         Transport for Configured Subscriptions";
    }

    identity grpc {
      base transport-protocol;
```

```
    description
      "Identity for grpc as transport protocol.";
  }

  identity security-protocol {
    description
      "Base identity for security protocol type.";
  }

  identity tls {
    base security-protocol;
    description
      "Identity for tls security protocol.";
  }

  identity dtls {
    base security-protocol;
    description
      "Identity for dtls security protocol.";
  }

  identity ssh {
    base security-protocol;
    description
      "Identity for ssh transport protocol.";
  }

  identity encoding-format {
    description
      "Base identity for encoding format type.";
  }

  identity xml {
    base encoding-format;
    description
      "Identity for xml encoding format.";
  }

  identity json {
    base encoding-format;
    description
      "Identity for json encoding format.";
  }

  identity binary {
    base encoding-format;
    description
      "Identity for binary encoding format.";
```

```
     }

     identity cbor {
       base binary;
       description
         "Identity for cbor encoding format.";
     }

     augment "/sysc:system-capabilities" {
       description
         "Add system level capability.";
       container data-export-capabilities {
         description
           "Capabilities related to data export notification capabilities negotia
tion.";
         list data-export-capability {
           description
             "Capability list related to data export notification capabilities ne
gotiation.";
           leaf transport-protocol {
             type identityref {
               base transport-protocol;
             }
             description
               "Type of transport protocol.";
           }
           leaf-list encoding-format {
             type identityref {
               base encoding-format;
             }
             description
               "Type of encoding format.";
           }
           leaf security-protocol {
             type identityref {
               base security-protocol;
             }
             description
               "Type of secure transport.";
           }
           leaf message-bundling-support {
             type empty;
             description
               "Enables message bundling support.";
           }
         }
       }
     }
   }
   <CODE ENDS>
```

4. IANA Considerations

4.1. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

```
URI:
   urn:ietf:params:xml:ns:yang:ietf-data-export-capabilities
Registrant Contact:
   The IESG.
XML:
   N/A; the requested URI is an XML namespace.
```

4.2. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
name:
   ietf-data-export-capabilities
namespace:
   urn:ietf:params:xml:ns:yang:ietf-data-export-capabilities
prefix:
   dec
reference:
   RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
   this note.)
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All protocol-accessible data nodes are read-only and cannot be modified. The data in these modules is not security sensitive. Access control may be configured, to avoid exposing the read-only data.

When this data is in file format, data should be protected against modification or unauthorized access using normal file handling mechanisms.

6. Contributors

Ran Tao
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China
Email: taoran20@huawei.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

Thomas Graf
Swisscom
Binzring 17
Zuerich 8045
Switzerland

Email: thomas.graf@swisscom.com

7. References

7.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

[RFC9196] Lengyel, B., Clemm, A., and B. Claise, "YANG Modules Describing Capabilities for Systems and Datastore Update Notifications", RFC 9196, DOI 10.17487/RFC9196, February 2022, <https://www.rfc-editor.org/info/rfc9196>.

7.2. Informative References

[I-D.ietf-netconf-https-notif]
Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for YANG Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-https-notif-09, 24 October 2021, <https://www.ietf.org/archive/id/draft-ietf-netconf-https-notif-09.txt>.

[I-D.ietf-netconf-notification-messages]
Voit, E., Jenkins, T., Birkholz, H., Bierman, A., and A. Clemm, "Notification Message Headers and Bundles", Work in Progress, Internet-Draft, draft-ietf-netconf-notification-messages-08, 17 November 2019, <https://www.ietf.org/archive/id/draft-ietf-netconf-notification-messages-08.txt>.

[I-D.ietf-netconf-udp-notif]
Zheng, G., Zhou, T., Graf, T., Francois, P., Feng, A. H., and P. Lucente, "UDP-based Transport for Configured Subscriptions", Work in Progress, Internet-Draft, draft-ietf-netconf-udp-notif-04, 21 October 2021, <https://www.ietf.org/archive/id/draft-ietf-netconf-udp-notif-04.txt>.

[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>.

[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>.

Appendix A. Usage Example of interaction with UDP Notif and HTTP Notif for Configured Subscription

The following instance-data example describes the Data Export Notification capabilities of a hypothetical "acme-router".

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=
"urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Server Capability Discovery</description>
  <content-data>
    <system-capabilities
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities"
      xmlns:inc="urn:ietf:params:xml:ns:yang:ietf-notification-capabilities"
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <data-export-capabilities>
        <data-export-capability>
          <transport-protocol>http-notif</transport-protocol>
          <encoding-format>json</encoding-format>
          <encoding-format>xml</encoding-format>
        </data-export-capability>
        <data-export-capability>
          <transport-protocol>udp-notif</transport-protocol>
          <encoding-format>binary</encoding-format>
        </data-export-capability>
      </data-export-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

In addition, the client could also query data export capability from the server. For example, the client sends <get> request message to the the server to query data export capability from the server.

```
 <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
     <filter type="subtree">
     <system-capabilities xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabili
ties">
       <data-export-capabilities/>
     </system-capabilities>
    </filter>
   </get>
 </rpc>
```

The server returns server data export capability using <rpc-reply> as follows:

```
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
  <system-capabilities
   xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities"
   xmlns:dec="urn:ietf:params:xml:ns:yang:ietf-data-export-capabilities">
    <data-export-capabilities>
      <data-export-capability>
        <transport-protocol>http-notif</transport-protocol>
        <encoding-format>json</encoding-format>
      </data-export-capability>
      <data-export-capability>
        <transport-protocol>udp-notif</transport-protocol>
        <encoding-format>binary</encoding-format>
      </data-export-capability>
    </data-export-capabilities>
  </system-capabilities>
</data>
</rpc-reply>
```

Appendix B. Changes between Revisions

v06-v07

* Delete the per-node related capability parameters from the Appendix.

v05-v06

* Revise abstract and introduction sessions so that the scope of this draft is not limited to telemetry but other notification.

* Revise the description of module ietf-system-capabilities to align with the latest version of draft-ietf-netconf-notification-capabilities.

* Remove compression-mode, timer-event-support and suppress-redundant parameters in the model.

* Move per-node related capability parameters to appendix section.

* Add a container to wrap data export capabilities to cleanly separate different groups of capabilities.

v04 - v05

* Change per-node-capabilities related parameters into empty type.

* Revise abstract and introduction section to only focus on capability fetching mechanism from the client to the server.

* Update Usage Example of interaction with HTTP-Notif and UDP-Notif for Configured Subscription.

v03 - v04

* Add interface namespace in the Adaptive Subscription usage example.

* subtrees and data nodes changes in the security section.

* Two compression mode related identities change.

* Move message-bundling-support parameer to system capabilities level.

* Add an example to discuss report reciever capability from the client per yang instance file format.

* Change encoding format from leaf to leaf-list and support multiple encoding formats for the same transport specific notif.

v02 - v03

* Change 'data-export-capabilities' into list type to support multiple transport protocol, encoding on the server.

* Add Usage Example of interaction with UDP based Transport for Configured Subscription.

* Add Thomas Graf as a contributor;

* Update motivation in the introduction to clarify why this work is needed.

* Support udp notif and http notif as two optional transport in the YANG data model.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: bill.wu@huawei.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

Peng Liu
China Mobile
Beiqijia Town, Changping District
Beijing
Email: liupengyjy@chinamobile.com

Wei Wang
China Telecom
32 Xuanwumen West St, Xicheng District
Beijing
Email: wangw36@chinatelecom.cn

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2021

Q. Wu
B. Wu
Huawei
P. Liu
H. Cai
China Mobile

July 8, 2020

Self-explanation data Node tag capability
draft-tao-netconf-notif-node-tag-capabilities-02

Abstract

Before a client application subscribes to updates from a datastore, server capabilities related to "Subscription to YANG Datastores" can be advertised using YANG Instance Data format. These server capabilities can be documented at implement time or reported at run-time.

This document proposes a YANG module for self-explanation data Node tag capability which augments system capabilities model and provide additional self-explanation data node attributes associated with node selectors within per-node capabilities.

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 January 9, 2021.

Copyright Notice

Table of Contents

1. Introduction

As described in [I-D.netconf-notification-capabilities], a server supporting YANG-Push MAY have a number of capabilities such as

- Supported (reporting) periods for periodic subscriptions;
- Maximum number of objects that can be sent in an update;
- Supported dampening periods for on-change subscriptions;
- The set of data nodes for which on-change notification is supported.

Notification capability model defined in [I-D.netconf-notification-capabilities] allows a client to discover basic system capability and YANG-Push related capabilities both at implementation-time and run-time. Without using this notification capability, it might lead to unexpected failures or additional message exchanges for NETCONF

clients to discover data objects with specific capability supported by a NETCONF server.

When all telemetry data on the server subscribed by a particular subscriber is huge, it becomes more likely that a burst of streamed data may temporarily overwhelm a receiver and consume expensive computing and storage resource. Accordingly, there is a need for filtering subscribed telemetry data on a server based on server capabilities, which can greatly reduce the amount of data to be streamed out to the destination.

However without telemetry data classification or prior knowledge of data objects correlation relationship, it is difficult for NETCONF clients to automatically select target data objects that are of interest to the client applications, e.g., identify a set of objects from different YANG data modules which have a common characteristic, collect specific object type nodes for multiple dimensional network visibility analysis.

This document proposes a YANG module for self-explanation data Node tag capability which augments System Capabilities model and provide additional self-explanation data node tag attributes associated with node selector for queries filtering.

1.1. Terminology

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.

2. Self-explanation data Node tag capability

The YANG module ietf-notification-capabilities defined in [I-D.netconf-notification-capabilities] specifies the following server capabilities related to YANG Push:

- A set of capabilities related to the amount of notifications the server can send out
- Specification of which data nodes support on-change notifications.
- Capability values can be specified on server level, datastore level or on specific data nodes (and their contained sub-tree) of a specific datastore. Capability values on a smaller, more specific part of the server's data always override more generic values.

- On-change capability is not specified on a server level as different datastores usually have different on-change capabilities. On a datastore level on-change capability for configuration and state data can be specified separately.

These server capabilities can be provided either at implementation time or reported at run time.

This document augments system capabilities model and provide additional data node self explaination tag attributes associated with node selector within per-node capabilities:

- Specification of which object type nodes, which performance metric nodes, which property related nodes they can push to the target recipient;

- Specification of measurement precision or granularity associated with performance metric related data nodes;

- Specification of operation type associated with performance metric related data nodes;

- Specification of service classification information associated with data nodes;

- Specification of task group information associated with a set of data nodes;

- Specification of self tag name of a set of data nodes they can push to the target recipient.

- Specification of data source type associated with a set of data nodes;

- Specfication of multi-source aggregation assocaited with a set of data nodes;

2.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```
    module: ietf-self-explanation-capabilities
    augment /sysc:system-capabilities/sysc:datastore-capabilities/ +
          sysc:per-node-capabilities/sys:node-selection/sys:node-selector:
       +--ro self-describing-attributes* [self-tag-id]
           +--ro self-tag-id             string
           +--ro opm-tag                 tags:tag
           +--ro metric-precision        tags:tag
           +--ro metric-scale            tags:tag
           +--ro operation-type          tags:tag
           +--ro service-tag*            tags:tag
           +--ro task-tag*               tags:tag
           +--ro data-source             tags:tag
           +--ro muli-source-tag         tags:tag
```

3. YANG Module

```
<CODE BEGINS> file "ietf-self-explanation-capabilities.yang"
module ietf-self-explanation-capabilities {
  yang-version 1.1;
  namespace urn:ietf:params:xml:ns:yang:ietf-self-explanation-capabilities;
  prefix sec;
  import ietf-system-capabilities  { prefix sysc ; }
  import ietf-module-tags  { prefix tags; }
  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Editor:   Qin Wu
               <mailto:bill.wu@huawei.com>
               Bo Wu
               <mailto:lana.wubo@huawei.com";
  description
    "This module defines an extension to System Capabilities model
     and provides additional self explaination data node tag attributes
     associated with node selector for queries filtering.

     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
     (http://trustee.ietf.org/license-info).
```

```
     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.";
         revision 2020-07-08 {
    description
      "Initial revision";
    reference
      "RFC XXXX";
  }
    augment "/sysc:system-capabilities/sysc:datastore-capabilities" +
         "/sysc:per-node-capabilities" +
          "/sysc:node-selection/sysc:node-selector" {
     description "Allows per-node capabilities have additional self-explanation a
ttributes";
     list self-describing-attributes {
      key self-tag-id;
      description "self describing attributes for each data node.";
       leaf self-tag-id {
         type string;
         description
          "This self tag id is used to uniquely identify a set of data nodes
          of the same group which have a common characteristic. If the opm-tag
          is metric, self-tag-id represents specific metric name. If the opm-tag
          is metric-group, self-tag-id represents specific metric group name.";
       }
         leaf opm-tag {
           type tags:tag ;
           description
            "Object, Property and Metric(OPM) Tags associated with
            the data node within YANG module.
            See the IANA 'YANG Data Node Tag Prefixes' registry
            for reserved prefixes and the IANA
            'IETF YANG Data Node Tags' registry for IETF tags.";
       }
         leaf metric-precision {
           type tags:tag;
           description
             "The numeric expression precision of performance
              metric related data node.";
         }

        leaf metric-scale {
            type tags:tag;
            description
             "The measurement scale of performance
              metric related data node.";
         }
         leaf operation-type{
          type tags:tag;
          description
```

```
            "Statistics operation of performance metric related
            data node.If the operation type is threshold type, the corresponding
            data object support threshold handling,e.g.,scan all interfaces
             for a certain type every 5 seconds and check the counters or
            status to cross threshold, return an array of interface entries
            that match the search.If the operation type is average,min,max, sum,
            it indicate the data object supports statistics operation, e.g.,
            scan all interfaces for a certain type every 5 seconds up to 60 second
s,
            only return min, average, max, sum value of specific data object rathe
r than
            the values that are current at the end of 60 seconds.";
          }
          leaf-list service-tag {
           type tags:tag;
           description
            "The node-service-tag can be used to provide a service
            classification information (e.g., tunnel, l3vpn,l2vpn)
            information associated with YANG data node.";
          }
          leaf-list task-tag {
           type tags:tag;
           description
            "The node-task-tag can be used to provide a task
            classification information (e.g., fault management,
            performance measurement) information associated with
            YANG data node.";
          }
          leaf data-source {
           type tags:tag;
           description
            "The data source type can be used to identify different
             data source type(e.g., service flow, resource, policy,
             qos, hardware).";
          }
          leaf multi-source-tag {
           type tags:tag;
           description
            "The multiple source tag can be used to aggregate peformance
             metric from different sources.";
          }
      }
    }
}
<CODE ENDS>
```

4. IANA Considerations

4.1. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

```
URI:
   urn:ietf:params:xml:ns:yang:ietf-self-explanation-capabilities
Registrant Contact:
   The IESG.
XML:
   N/A; the requested URI is an XML namespace.
```

4.2. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
name:
   ietf-self-explanation-capabilities
namespace:
   urn:ietf:params:xml:ns:yang:ietf-self-explanation-capabilities
prefix:
   sec
reference:
   RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
   this note.)
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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 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:

- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:self-tag-id
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:opm-tag
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:metric-precision
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:operation-type
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:service-tag
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:task-tag
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:data-source
- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:multi-source-tag

6. Contributors

The authors would like to thank Ran Tao for his major contributions to the initial modeling and use cases.

7. References

7.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

7.2. Informative References

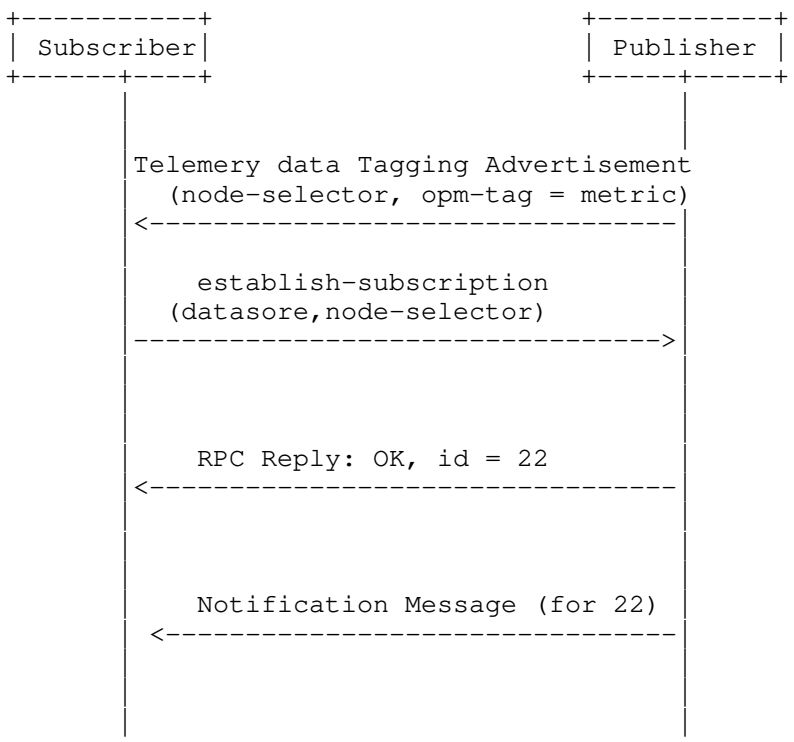
[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[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>.

Appendix A. Targeted data object subscription example

The following subsections provides targeted data object subscription example.The subscription "id" values of 22 used below is just an example. In production, the actual values of "id" might not be small integers.



The publisher advertise telemetry data node capability to the subscriber to instruct the receiver to subscribe targeted data object with specific characteristics (e.g., performance metric related data object) and specific data path corresponding to the targeted data object.

The following XML example [W3C.REC-xml-20081126] illustrates the advertisment of the list of available target objects:

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=\
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Defines the notification capabilities of an acme-router.
    The router only has running, and operational datastores.
    Every change can be reported on-change from running, but
    only config=true nodes and some config=false data from operational.
    Statistics are not reported based on timer based trigger and counter
    threshold based trigger.
  </description>
  <content-data>
    <system-capabilities \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities" \
      xmlns:inc=\
        "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities" \
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics/if:in-errors\
          </node-selector>
          <sec:self-describing-capabilities>
            <sec:self-tag-id>bandwith</sec:self-tag-id>
            <sec:opm-tag>metric</sec:opm-tag>
          </sec:self-describing-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

With telemetry data tagging information carried in the Telemetry data Tagging Advertisement, the subscriber identifies targeted data object and associated data path to the datastore node and sends a establish-subscription RPC to subscribe specific data objects that are interests to the client application from the publisher.

```
 <netconf:rpc message-id="101"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
          xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:operational
     </yp:datastore>
     <yp:datastore-xpath-filter
         xmlns:ex="https://example.com/sample-data/1.0">
       /if:interfaces/if:interface/if:statistics/if:in-errors
     </yp:datastore-xpath-filter>
     <yp:periodic>
       <yp:period>500</yp:period>
     </yp:periodic>
   </establish-subscription>
 </netconf:rpc>
```

The publisher returns specific object type of operational state related to the subscriber.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Bo Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: lana.wubo@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

Hui Cai
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: caihui@chinamobile.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2021

Q. Wu
Q. Ma
Huawei
P. Liu
Y. Fu
China Mobile

October 29, 2020

Self describing data Node tag capability
draft-tao-netconf-notif-node-tag-capabilities-03

Abstract

Before a client application subscribes to updates from a datastore, server capabilities related to "Subscription to YANG Datastores" can be advertised using YANG Instance Data format. These server capabilities can be documented at implement time or reported at run-time.

This document proposes a YANG module for self describing data Object tag capability which augments system capabilities model and provide additional self describing data node attributes associated with node selectors within per-node capabilities.

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 May 2, 2021.

Copyright Notice

Table of Contents

1. Introduction

As described in [I-D.netconf-notification-capabilities], a server supporting YANG-Push MAY have a number of capabilities such as

- Supported (reporting) periods for periodic subscriptions;
- Maximum number of objects that can be sent in an update;
- Supported dampening periods for on-change subscriptions;
- The set of data nodes for which on-change notification is supported.

Notification capability model defined in [I-D.netconf-notification-capabilities] allows a client to discover basic system capability and YANG-Push related capabilities both at implementation-time and run-time. Without using this notification capability, it might lead to unexpected failures or additional message exchanges for NETCONF

clients to discover data objects with specific capability supported by a NETCONF server.

When all telemetry data on the server subscribed by a particular subscriber is huge, it becomes more likely that a burst of streamed data may temporarily overwhelm a receiver and consume expensive computing and storage resource. Accordingly, there is a need for filtering subscribed telemetry data on a server based on server capabilities, which can greatly reduce the amount of data to be streamed out to the destination.

However without telemetry data classification or prior knowledge of data objects correlation relationship, it is difficult for NETCONF clients to automatically select target data objects that are of interest to the client applications, e.g., identify a set of objects from different YANG data modules which have a common characteristic, collect specific object type nodes for multiple dimensional network visibility analysis.

This document proposes a YANG module for self describing data Node tag capability which augments System Capabilities model and provide additional self describing data node tag attributes associated with node selector for queries filtering.

1.1. Terminology

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.

2. Self-explanation data Node tag capability

The YANG module ietf-notification-capabilities defined in [I-D.netconf-notification-capabilities] specifies the following server capabilities related to YANG Push:

- A set of capabilities related to the amount of notifications the server can send out
- Specification of which data nodes support on-change notifications.
- Capability values can be specified on server level, datastore level or on specific data nodes (and their contained sub-tree) of a specific datastore. Capability values on a smaller, more specific part of the server’s data always override more generic values.

- On-change capability is not specified on a server level as different datastores usually have different on-change capabilities. On a datastore level on-change capability for configuration and state data can be specified separately.

These server capabilities can be provided either at implementation time or reported at run time.

This document augments system capabilities model and provide additional data node self explaination tag attributes associated with node selector within per-node capabilities:

- Specification of which data objects (e.g., data object tagged with object tag, property subobject tag, metri subobject tag) they can push to the target recipient;

- Specification of metric group tag associated with a set of metric subobjects;

- Specfication of multi-source aggregation tag assocaited with specific metric subobject;

2.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```
 module: ietf-self-describing-capabilities
 augment /sysc:system-capabilities/sysc:datastore-capabilities/ +
       sysc:per-node-capabilities/sys:node-selection/sys:node-selector:
    +--ro self-describing-attributes
        +--ro opm-tag*                 tags:tag
        +--ro metric-group            tags:tag
        +--ro muli-source-tag         tags:tag
```

3. YANG Module

```
<CODE BEGINS> file "ietf-self-describing-capabilities.yang"
module ietf-self-describing-capabilities {
  yang-version 1.1;
  namespace urn:ietf:params:xml:ns:yang:ietf-self-description-capabilities;
  prefix sdc;
  import ietf-system-capabilities  { prefix sysc ; }
  import ietf-module-tags  { prefix tags; }
  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://tools.ietf.org/wg/netconf/>
```

```
     WG List:  <mailto:netconf@ietf.org>

     Editor:   Qin Wu
               <mailto:bill.wu@huawei.com>
               Qiufang Ma
               <mailto:maqiufang1@huawei.com>
               Peng Liu
               <liupengyjy@chinamobile.com>
               Hui Cai
               <caihui@chinamobile.com>";
  description
    "This module defines an extension to System Capabilities model
     and provides additional self explaination data node tag attributes
     associated with node selector for queries filtering.

     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
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;
     see the RFC itself for full legal notices.";
    revision 2020-07-08 {
    description
      "Initial revision";
    reference
      "RFC XXXX";
  }
    augment "/sysc:system-capabilities/sysc:datastore-capabilities" +
         "/sysc:per-node-capabilities" +
          "/sysc:node-selection/sysc:node-selector" {
     description "Allows per-node capabilities have additional self-explanation a
ttributes";
     container self-describing-attributes {
      description "self describing attributes for specific data node.";
         leaf-list opm-tag {
           type tags:tag;
           description
            "Object, Property and Metric(OPM) Tags associated with
             specific data object within YANG module.
            See the IANA 'YANG Data Node Tag Prefixes' registry
            for reserved prefixes and the IANA
            'IETF YANG Data Node Tags' registry for IETF tags.";
         }
```

```
          leaf metric-group {
            type tags:tag;
            description
            "The metric-group can be used to provide correlation between
            different performance metric information associated with YANG
            data node.";
          }
          leaf multi-source-tag {
           type tags:tag;
           description
            "The multiple source tag can be used to aggregate peformance
             metric from different sources.";
          }
      }
    }
}
<CODE ENDS>
```

4. IANA Considerations

4.1. Updates to the IETF XML Registry

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in [RFC3688], the following registration has been made:

```
URI:
   urn:ietf:params:xml:ns:yang:ietf-self-describing-capabilities
Registrant Contact:
   The IESG.
XML:
   N/A; the requested URI is an XML namespace.
```

4.2. Updates to the YANG Module Names Registry

This document registers one YANG module in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registration has been made:

```
name:
   ietf-self-describing-capabilities
namespace:
   urn:ietf:params:xml:ns:yang:ietf-self-describing-capabilities
prefix:
   sec
reference:
   RFC XXXX (RFC Ed.: replace XXX with actual RFC number and remove
   this note.)
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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 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:

- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:opm-tag

- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:metric-group

- /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities/sys:node-selection/sys:node-selector/sec:self-describing-attributes/sec:multi-source-tag

6. Contributors

The authors would like to thank Ran Tao, Hui Cai for his major contributions to the initial modeling and use cases.

7. References

7.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

7.2. Informative References

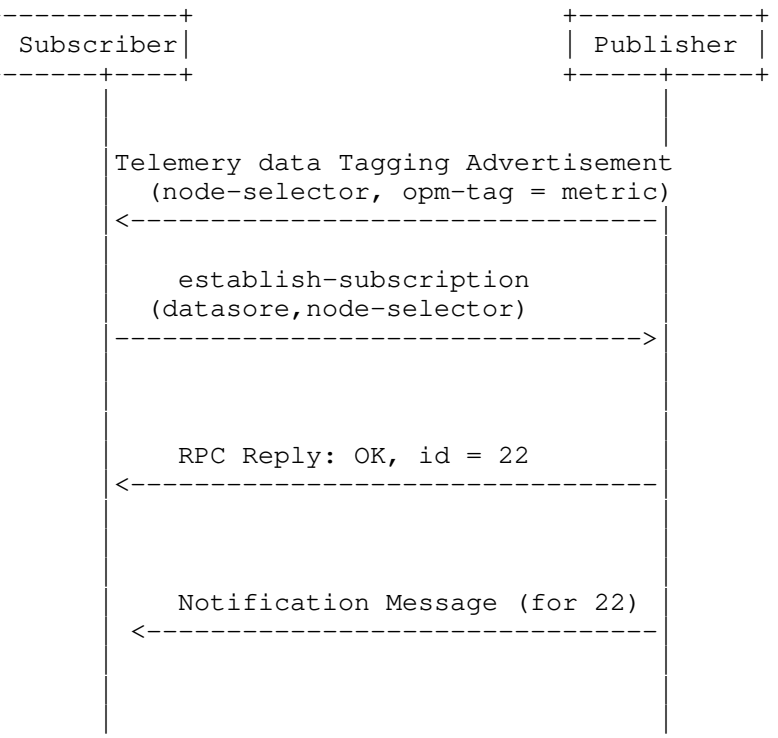
[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[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>.

Appendix A. Targeted data object subscription example

The following subsections provides targeted data object subscription example. The subscription "id" values of 22 used below is just an example. In production, the actual values of "id" might not be small integers.



The publisher advertise telemetry data node capability to the subscriber to instruct the receiver to subscribe targeted data object with specific characteristics (e.g., performance metric related data object) and specific data path corresponding to the targeted data object.

The following XML example [W3C.REC-xml-20081126] illustrates the advertisment of the list of available target objects:

```
<?xml version="1.0" encoding="UTF-8"?>
<instance-data-set xmlns=\
    "urn:ietf:params:xml:ns:yang:ietf-yang-instance-data">
  <name>acme-router-notification-capabilities</name>
  <content-schema>
    <module>ietf-system-capabilities@2020-03-23</module>
    <module>ietf-notification-capabilities@2020-03-23</module>
    <module>ietf-data-export-capabilities@2020-03-23</module>
  </content-schema>
  <!-- revision date, contact, etc. -->
  <description>Defines the notification capabilities of an acme-router.
    The router only has running, and operational datastores.
    Every change can be reported on-change from running, but
    only config=true nodes and some config=false data from operational.
    Statistics are not reported based on timer based trigger and counter
    threshold based trigger.
  </description>
  <content-data>
    <system-capabilities \
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities" \
      xmlns:inc=\
        "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities" \
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      <datastore-capabilities>
        <datastore>ds:operational</datastore>
        <per-node-capabilities>
          <node-selector>\
              /if:interfaces/if:interface/if:statistics/if:in-errors\
          </node-selector>
          <sec:self-describing-capabilities>
            <sec:opm-tag>metric</sec:opm-tag>
            <sec:metric-group>loss</sec:metric-group>
          </sec:self-describing-capabilities>
        </per-node-capabilities>
      </datastore-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

With telemetry data tagging information carried in the Telemetry data Tagging Advertisement, the subscriber identifies targeted data object and associated data path to the datastore node and sends a establish-subscription RPC to subscribe specific data objects that are interests to the client application from the publisher.

```
<netconf:rpc message-id="101"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
      xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <yp:datastore
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
      /if:interfaces/if:interface/if:statistics/if:in-errors
    </yp:datastore-xpath-filter>
    <yp:periodic>
      <yp:period>500</yp:period>
    </yp:periodic>
  </establish-subscription>
</netconf:rpc>
```

The publisher returns specific object type of operational state related to the subscriber.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: maqiufang1@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

Yuexia Fu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: yuexiafu@chinamobile.com

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2021

T. Zhou
G. Zheng
Huawei
E. Voit
Cisco Systems
T. Graf
Swisscom
P. Francois
INSA-Lyon

July 10, 2020

Subscription to Distributed Notifications
draft-unyte-netconf-distributed-notif-00

Abstract

This documents describes extensions to the YANG notifications subscription to allow metrics being published directly from processors on line cards to target receivers, while subscription is still maintained at the route processor in a distributed forwarding system.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 January 11, 2021.

Table of Contents

1. Introduction

The mechanism to support a subscription to a continuous and customized stream of updates from a YANG datastore is defined in [RFC8639] and [RFC8641]. Requirements for Subscription to YANG Datastores are defined in [RFC7923]

By streaming data from publishers to receivers, much better performance and fine-grained sampling can be achieved than with polling. In a distributed forwarding system, the packet forwarding

is delegated to multiple processors on line cards. To not to overhelm the route processor resources, it is not uncomon that data records are published directly from processors on line cards to target Receivers to further increase efficency on the routing system.

This documents complement the general subscription requirements defined in section 4.2.1 of [RFC7923] by the paragraph: A Subscription Service MAY support the ability to export from multiple software processes on a single routing system and expose the information which software process produced which message to maintain data integrity.

2. Terminologies

The following terms are defined in [RFC8639] and are not redefined here:

Subscriber

Publisher

Receiver

Subscription

In addition, this document defines the following terms:

Global Subscription: the Subscription requested by the subscriber. It may be decomposed into multiple Component Subscriptions.

Component Subscription: is the Subscription that defines a data source which is managed and controlled by a single Publisher.

Global Capability: is the overall subscription capability that the group of Publishers can expose to the Subscriber.

Component Capability: is the subscription capability that each Publisher can expose to the Subscriber.

Master: is the Publisher that interacts with the Subscriber to deal with the Global Subscription. It decomposes the Global Subscription to multiple Component Subscriptions and interacts with the Agents.

Agent: is the Publisher that interacts with the Master to deal with the Component Subscription and pushing the data to the collector.

3. Solution Overview

Figure 2 below shows the distributed data export framework.

A collector usually includes two components,

- the Subscriber generates the subscription instructions to express what and how the collector want to receive the data;
- the Receiver is the target for the data publication.

For one subscription, there are one or more Receivers. And the Subscriber does not necessarily share the same IP address as the Receivers.

In this framework, the Publisher pushes data to the Receiver according to the subscription. The Publisher is either in the Master or Agent role. The Master knows all the capabilities that his Agents are able to provide and exposes the Global Capability to the collector. The Subscriber maintains the Global Subscription at the Master and disassembles the Global Subscription to multiple Component Subscriptions, depending from which source data is needed. The Component Subscriptions are then distributed to the corresponding Publisher Agents on route and processors on line cards.

Publisher Agents collects metrics according to the Component Subscription, add its metadata, encapsulate and pushes data to the Receiver where packets are reassembled and decapsulated.

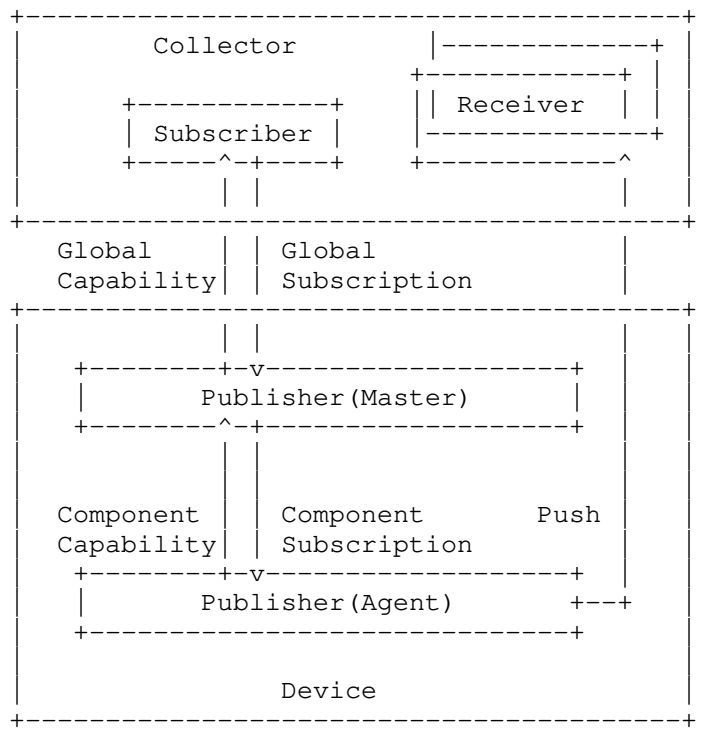


Fig. 2 The Distributed Data Export Framework

Master and Agents interact with each other in several ways:

- Agents need to register at the Master at the beginning of their process life-cycle

- Contracts are created between the Master and each Agent on the Component Capability, and the format for streaming data structure.

- The Master relays the component subscriptions to the Agents.

- The Agents announce the status of their Component Subscriptions to the Master. The status of the overall subscription is maintained by the Master. The Master is responsible for notifying the subscriber in case of problems with the Component Subscriptions.

The technical mechanisms or protocols used for the coordination of operational information between Master and Agent is out-of-scope of this document.

4. Subscription Decomposition

The Collector can only subscribe to the Master. This requires the Master to:

1. expose the Global Capability that can be served by multiple Publisher Agents;

2. disassemble the Global Subscription to multiple Component Subscriptions, and distribute them to the Publisher Agents of the corresponding metric sources so that they not overlap;

3. notify on changes when portions of a subscription moving between different Publisher Agents over time.

And the Agent to:

- Inherit the Global Subscription properties from Publisher Master for its Component Subscription;

- share the same life-cycle as the Global Subscription;

- share the same Subscription ID as the Global Subscription.

5. Publication Composition

The Publisher Agent collects data and encapsulates the packets per Component Subscription. The format and structure of the data records are defined by the YANG schema, so that the decomposition at the Receiver can benefit from the structured and hierarchical data records.

The Receiver is able to associate the YANG data records with Subscription ID [RFC8639] to the subscribed subscription and with Message Generator ID [I-D.ietf-netconf-notification-messages] to one of the Publisher Agents software processes to enable message integrity.

For the dynamic subscription, the output of the "establish-subscription" RPC defined in [RFC8639] MUST include a list of Message Generator IDs to indicate how the Global Subscription is decomposed into several Component Subscriptions.

The "subscription-started" and "subscription-modified" notification defined in [RFC8639] MUST also include a list of Message Generator IDs to notify the current Publishers for the corresponding Global Subscription.

6. Subscription State Change Notifications

In addition to sending event records to Receivers, the Master MUST also send subscription state change notifications [RFC8639] when events related to subscription management have occurred. All the subscription state change notifications MUST be delivered by the Master.

When the subscription decomposition result changed, the "subscription-modified" notification MUST be sent to indicate the new list of Publishers.

7. Publisher Configurations

This document assumes that all Publisher Agents are preconfigured to push data. The actual working Publisher Agents are selected based on the subscription decomposition result.

All Publisher Agents share the same source IP address for data export. For connectionless data transport such as UDP based transport [I-D.unyte-netconf-udp-notif] the same Layer 4 source port for data export can be used. For connection based data transport such as HTTPS based transport [I-D.ietf-netconf-https-notif], each Publisher Agent MUST be able to acknowledge packet retrieval from Receivers, and therefore requires a dedicated Layer 4 source port per software process.

The specific configuration on transports is described in the resposible documents.

8. YANG Tree

```
module: ietf-distributed-notifications
  augment /sn:subscriptions/sn:subscription:
    +--ro message-generator-id*   string
  augment /sn:subscription-started:
    +--ro message-generator-id*   string
  augment /sn:subscription-modified:
    +--ro message-generator-id*   string
  augment /sn:establish-subscription/sn:output:
    +--ro message-generator-id*   string
```

9. YANG Module

```
<CODE BEGINS> file "ietf-distributed-notifications@2020-05-09.yang"
module ietf-distributed-notif {
  yang-version 1.1;
  namespace
```

```
    "urn:ietf:params:xml:ns:yang:ietf-distributed-notifications";
  prefix mso;
  import ietf-subscribed-notifications {
    prefix sn;
  }

  organization "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <http:/tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Editor:   Tianran Zhou
               <mailto:zhoutianran@huawei.com>

     Editor:   Guangying Zheng
               <mailto:zhengguangying@huawei.com>";


  description
    "Defines augmentation for ietf-subscribed-notifications to
    enable the distributed publication with single subscription.

    Copyright (c) 2018 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 XXXX; see the
    RFC itself for full legal notices.";

  revision 2020-05-09 {
    description
      "Initial version";
    reference
      "RFC XXXX: Subscription to Distributed Notifications";
  }

  grouping message-generator-ids {
    description
      "Provides a reusable list of message-generator-ids.";

    leaf-list message-generator-id {
      type string;
```

```
        config false;
        ordered-by user;
        description
          "Software process which created the message (e.g.,
           processor 1 on linecard 1). This field is
           used to  notify the collector the working originator.";
      }
    }

    augment "/sn:subscriptions/sn:subscription" {
      description
        "This augmentation allows the message generators to be
        exposed for a subscription.";

      uses message-generator-ids;
    }

    augment "/sn:subscription-started" {
      description
        "This augmentation allows MSO specific parameters to be
         exposed for a subscription.";

      uses message-generator-ids;
    }

    augment "/sn:subscription-modified" {
      description
        "This augmentation allows MSO specific parameters to be
         exposed for a subscription.";

      uses message-generator-ids;
    }

    augment "/sn:establish-subscription/sn:output" {
      description
        "This augmentation allows MSO specific parameters to be
         exposed for a subscription.";

      uses message-generator-ids;
    }
  }
  <CODE ENDS>
```

10. IANA Considerations

This document registers the following namespace URI in the IETF XML Registry [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications

Registrant Contact: The IESG.

XML: N/A; the requested URI is an XML namespace.

This document registers the following YANG module in the YANG Module Names registry [RFC3688]:

Name: ietf-subscribed-notifications

Namespace: urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications

Prefix: mso

Reference: RFC XXXX

11. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF Access Control Model (NACM) [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The new data nodes introduced in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get-config or notification) to this data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

o /subscriptions/subscription/message-generator-ids

The entries in the two lists above will show where subscribed resources might be located on the publishers. Access control MUST be set so that only someone with proper access permissions has the ability to access this resource.

Other Security Considerations is the same as those discussed in YANG-Push [RFC8641].

12. Contributors

Alexander Clemm
Futurewai
2330 Central Expressway
Santa Clara
California
United States of America
Email: ludwig@clemm.org

13. Acknowledgements

We thank Kent Watsen, Mahesh Jethanandani, Martin Bjorklund, Tim Carey and Qin Wu for their constructive suggestions for improving this document.

14. References

14.1. 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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <https://www.rfc-editor.org/info/rfc7923>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

14.2. Informative References

[I-D.ietf-netconf-https-notif]
Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for Configured Subscriptions", draft-ietf-netconf-https-notif-02 (work in progress), March 2020.

[I-D.ietf-netconf-notification-messages]
Voit, E., Jenkins, T., Birkholz, H., Bierman, A., and A. Clemm, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-08 (work in progress), November 2019.

[I-D.unyte-netconf-udp-notif]
Zhou, T., Zheng, G., Lucente, P., Graf, T., and P. Francois, "UDP-based Transport for Configured Subscriptions", draft-unyte-netconf-udp-notif-00 (work in progress), July 2020.

Appendix A. Examples

This appendix is non-normative.

A.1. Dynamic Subscription

Figure 3 shows a typical dynamic subscription to the device with distributed data export capability.

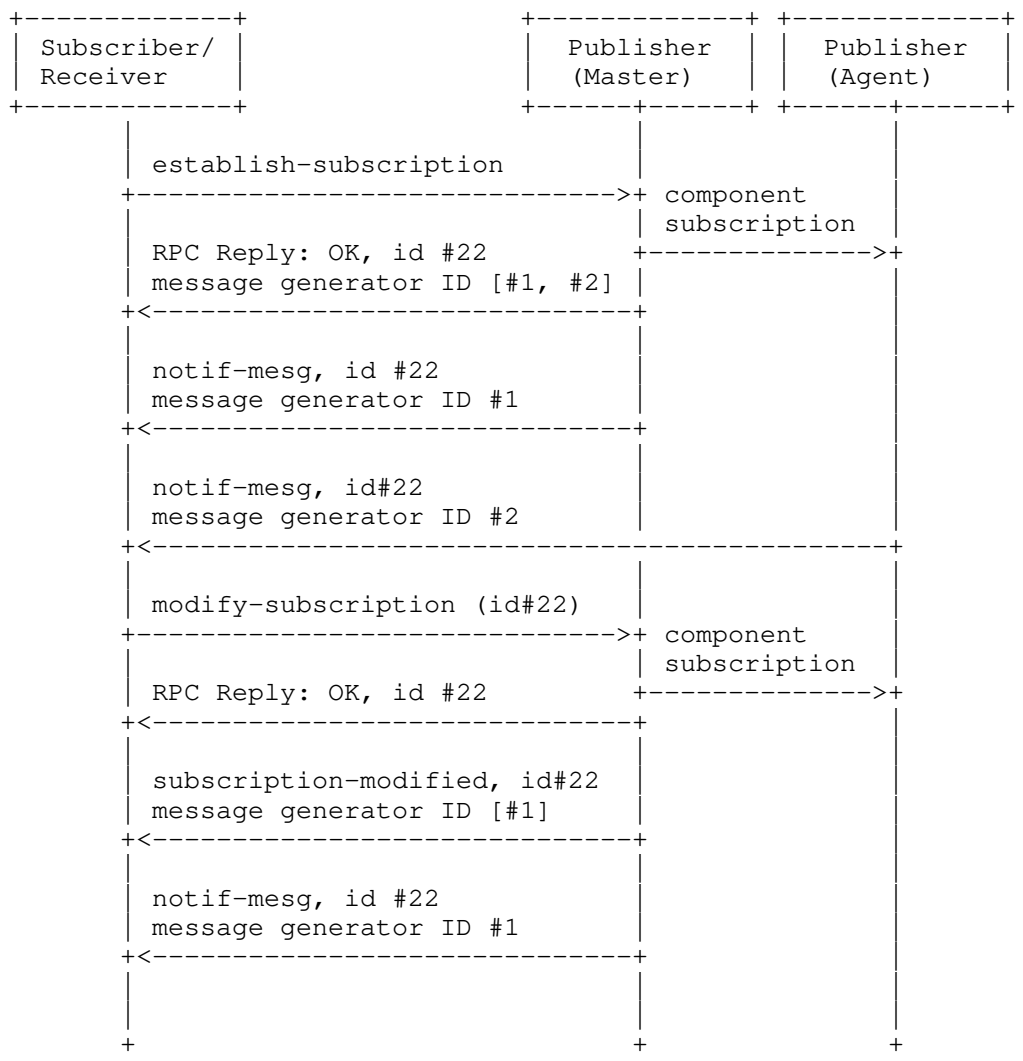


Fig. 3 Call Flow for Dynamic Subscription

A "establish-subscription" RPC request as per [RFC8641] is sent to the Master with a successful response. An example of using NETCONF:

```
<netconf:rpc message-id="101"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
          xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:operational
     </yp:datastore>
     <yp:datastore-xpath-filter
         xmlns:ex="https://example.com/sample-data/1.0">
       /ex:foo
     </yp:datastore-xpath-filter>
     <yp:periodic>
       <yp:period>500</yp:period>
     </yp:periodic>
   </establish-subscription>
 </netconf:rpc>
```

Fig. 4 "establish-subscription" Request

As the device is able to fully satisfy the request, the request is given a subscription ID of 22. The response as in Figure 5 indicates that the subscription is decomposed into two component subscriptions which will be published by two message generators: #1 and #2.

```
  <rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <id
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
        22
     </id>
     <message-generator-id
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
        1
     </message-generator-id>
     <message-generator-id
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
        2
     </message-generator-id>
  </rpc-reply>
```

Fig. 5 "establish-subscription" Positive RPC Response

Then, both Publishers send notifications with the corresponding piece of data to the Receiver.

The subscriber may invoke the "modify-subscription" RPC for a subscription it previously established. The RPC has no difference to the single publisher case as in [RFC8641]. Figure 6 provides an example where a subscriber attempts to modify the period and datastore XPath filter of a subscription using NETCONF.

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
      xmlns=
        "urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
      xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>22</id>
    <yp:datastore
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
        xmlns:ex="https://example.com/sample-data/1.0">
      /ex:bar
    </yp:datastore-xpath-filter>
    <yp:periodic>
      <yp:period>250</yp:period>
    </yp:periodic>
   </modify-subscription>
</rpc>
```

Fig. 6 "modify-subscription" Request

If the modification is successfully accepted, the "subscription-modified" subscription state notification is sent to the subscriber by the Master. The notification, Figure 7 for example, indicates the modified subscription is decomposed into one component subscription which will be published by message generator #1.

```
 <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2007-09-01T10:00:00Z</eventTime>
   <subscription-modified
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <id>22</id>
     <yp:datastore
         xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:operational
     </yp:datastore>
     <yp:datastore-xpath-filter
         xmlns:ex="https://example.com/sample-data/1.0">
       /ex:bar
     </yp:datastore-xpath-filter>
     <yp:periodic>
         <yp:period>250</yp:period>
     </yp:periodic>
     <message-generator-id
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notificationss>
        1
     </message-generator-id>
   </subscription-modified>
 </notification>
```

Fig. 7 "subscription-modified" Subscription State Notification

A.2. Configured Subscription

Figure 8 shows a typical configured subscription to the device with distributed data export capability.

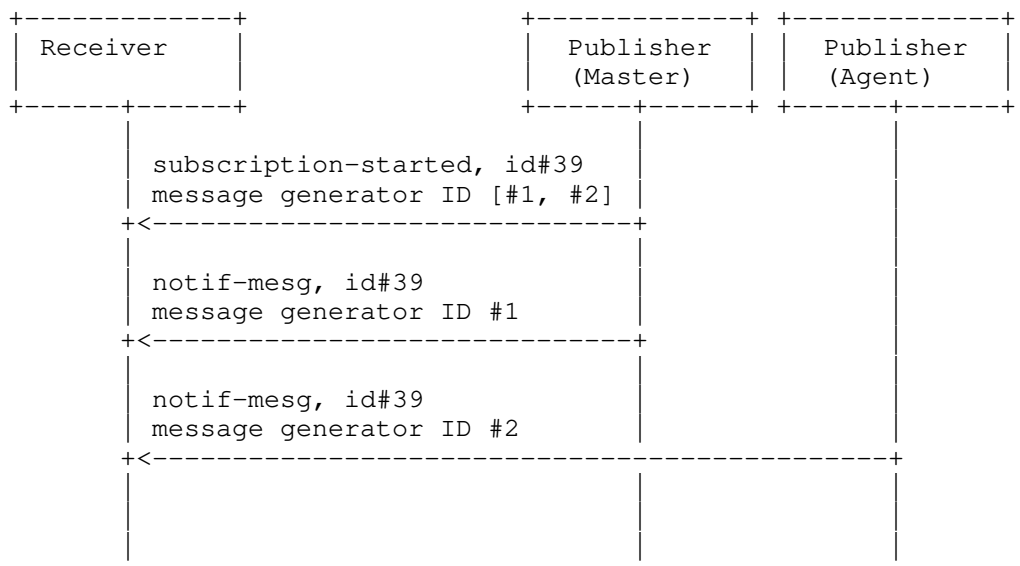


Fig. 8 Call Flow for Configured Subscription

Before starting to push data, the "subscription-started" subscription state notification is sent to the Receiver. The following example assumes the NETCONF transport has already established. The notification indicates that the configured subscription is decomposed into two component subscriptions which will be published by two message generators: #1 and #2.

```
 <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <eventTime>2007-09-01T10:00:00Z</eventTime>
   <subscription-started
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <identifier>39</identifier>
     <yp:datastore
         xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:operational
     </yp:datastore>
     <yp:datastore-xpath-filter
         xmlns:ex="https://example.com/sample-data/1.0">
       /ex:foo
     </yp:datastore-xpath-filter>
     <yp:periodic>
         <yp:period>250</yp:period>
     </yp:periodic>
     <message-generator-id
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
        1
     </message-generator-id>
     <message-generator-id
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
        2
     </message-generator-id>
   </subscription-started>
 </notification>
```

Fig. 9 "subscription-started" Subscription State Notification

Then, both Publishers send notifications with the corresponding data record to the Receiver.

Authors' Addresses

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Eric Voit
Cisco Systems
United States of America

Email: evoit@cisco.com

Thomas Graf
Swisscom
Binzring 17
Zuerich 8045
Switzerland

Email: thomas.graf@swisscom.com

Pierre Francois
INSA-Lyon
Lyon
France

Email: pierre.francois@insa-lyon.fr

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: January 11, 2021

G. Zheng
T. Zhou
Huawei
T. Graf
Swisscom
P. Francois
INSA-Lyon
P. Lucente
NTT

July 10, 2020

UDP-based Transport for Configured Subscriptions
draft-unyte-netconf-udp-notif-00

Abstract

This document describes an UDP-based notification mechanism to collect data from networking devices. A shim header is proposed to facilitate the streaming of data directly from line cards to a collector. The objective is to rely on a lightweight approach to allow for higher frequency and better transit performance compared to already established notification mechanisms.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 January 11, 2021.

Table of Contents

1. Introduction

Sub-Notif [RFC8639] defines a mechanism that lets a collector subscribe to the publication of YANG-defined data maintained in a YANG [RFC7950] datastore. The mechanism separates the management and control of subscriptions from the transport used to deliver the data. Three transport mechanisms, namely NETCONF transport [RFC8640], RESTCONF transport [RFC8650], and HTTPS transport [I-D.ietf-netconf-https-notif] have been defined so far for such notification messages.

While powerful in its features and general in their architecture, the currently available transport mechanisms need to be complemented to support data publications at high velocity from devices that feature a distributed architecture. The currently available transports are based on TCP and lack the efficiency needed to continuously send notifications at high velocity.

This document specifies a transport option for Sub-Notif that leverages UDP. Specifically, it facilitates the distributed data collection mechanism described in [I-D.unyte-netconf-distributed-notif]. In the case of data originating from multiple line cards, centralized designs require data to be internally forwarded from those line cards to the push server, presumably on a route processor, which then combines the individual data items into a single consolidated stream. The centralized data collection mechanism can result in a performance bottleneck, especially when large amounts of data are involved.

What is needed is the support for a mechanism that allows for directly pushing multiple substreams, e.g. one from each line card, without passing them through an additional processing stage for internal consolidation. The proposed UDP-based transport allows for such a distributed data collection approach.

- Firstly, a UDP approach reduces the burden of maintaining a large amount of active TCP connections at the collector, notably in cases where it collects data from the line cards of a large amount of networking devices.

- Secondly, as no connection state needs to be maintained, UDP encapsulation can be easily implemented by the hardware of the publication streamer, which will further improve performance.

- Ultimately, such advantages allow for a larger data analysis feature set, as more voluminous, finer grained data sets can be streamed to the collector.

The transport described in this document can be used for transmitting notification messages over both IPv4 and IPv6.

This document describes the notification mechanism. It is intended to be used in conjunction with [RFC8639], extended by [I-D.unyte-netconf-distributed-notif].

Section 2 describes the control of the proposed transport mechanism. Section 3 details the notification mechanism and message format. Section 4 discusses congestion control. Section 5 covers the applicability of the proposed mechanism.

2. Configured Subscription to UDP-Notif

This section describes how the proposed mechanism can be controlled using subscription channels based on NETCONF or RESTCONF.

Following the usual approach of Sub-Notif, configured subscriptions contain the location information of all the receivers, including the IP address and the port number, so that the publisher can actively send UDP-Notif messages to the corresponding receivers.

Note that receivers MAY NOT be already up and running when the configuration of the subscription takes effect on the monitored device. The first message MUST be a separate subscription-started notification to indicate the Receiver that the stream has started flowing. Then, the notifications can be sent immediately without delay. All the subscription state notifications, as defined in [RFC8639], MUST be encapsulated in separate notification messages.

3. UDP-Based Transport

In this section, we specify the UDP-Notif Transport behaviour. Section 3.1 describes the general design of the solution. Section 3.2 specifies the UDP-Notif message format. Section 3.3 describes a generic optional sub TLV format. Section 3.3.1 uses such options to provide a fragmentation solution for large UDP-Notif message payloads. Section 3.4 describes the encoding of the message payload.

3.1. Design Overview

As specified in Sub-Notif, the telemetry data is encapsulated in the NETCONF/RESTCONF notification message, which is then encapsulated and carried using transport protocols such as TLS or HTTP2. Figure 1 illustrates the the structure of an UDP-Notif message.

- The Message Header contains information that facilitate the message transmission before deserializing the notification message.

- Notification Message is the encoded content that the publication stream transports. The common encoding methods include GPB [1], CBOR [RFC7049], JSON, and XML. [I-D.ietf-netconf-notification-messages] describes the structure of the Notification Message for single notifications and bundled notifications.

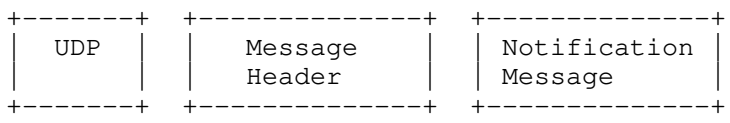


Figure 1: UDP-Notif Message Overview

3.2. Format of the UDP-Notif Message Header

The UDP-Notif Message Header contains information that facilitate the message transmission before deserializing the notification message. The data format is shown in Figure 2.

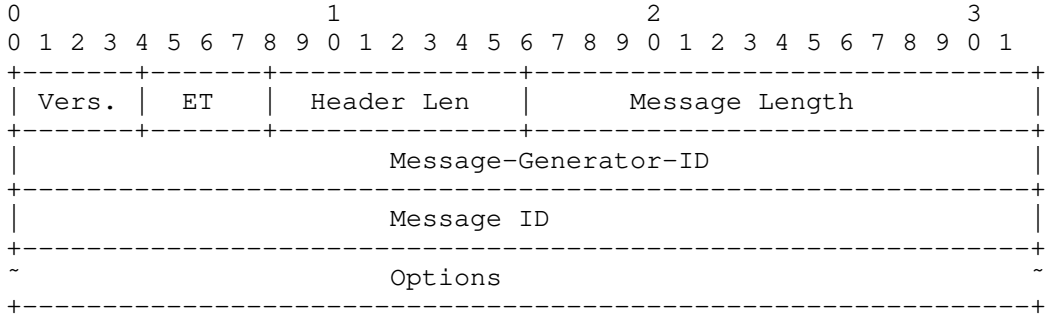


Figure 2: UDP-Notif Message Header Format

The Message Header contains the following field:

- Version represents the PDU (Protocol Data Unit) encoding version. The initial version value is 0.
- ET is a 4 bit identifier to indicate the encoding type used for the Notification Message. 16 types of encoding can be expressed:
  - 0: GPB;
  - 1: CBOR;
  - 2: JSON;
  - 3: XML;
  - others are reserved.
- Header Length is the length of the message header in octets, including both the fixed header and the options.

- Message Length is the total length of the message within one UDP datagram, measured in octets, including the message header.

- Message-Generator-ID is a 32-bit identifier of the process which created the notification message. This allows disambiguation of an information source, such as the identification of different line cards sending the notification messages. The source IP address of the UDP datagrams SHOULD NOT be interpreted as the identifier for the host that originated the UDP-Notif message. Indeed, the streamer sending the UDP-Notif message could be a relay for the actual source of data carried within UDP-Notif messages.

- The Message ID is generated continuously by the sender of UDP-Notif messages. Different subscribers share the same Message ID sequence.

- Options is a variable-length field in the TLV format. When the Header Length is larger than 12 octets, which is the length of the fixed header, Options TLVs follow directly after the fixed message header (i.e., Message ID). The details of the options are described in the following section.

3.3. Options

All the options are defined with the following format, illustrated in Figure 3.

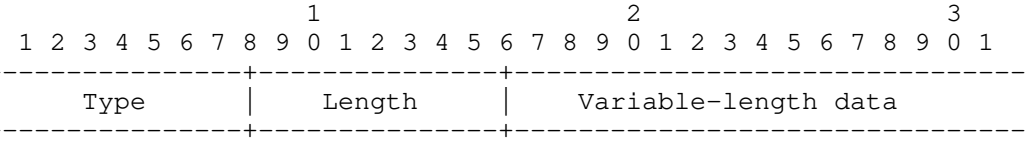


Figure 3: Generic Option Format

- Type: 1 octet describing the option type;

- Length: 1 octet of the TLV Length, including the Type and Length fields;

- Variable-length data: 0 or more octets of TLV Value.

3.3.1. Fragmentation Option

The UDP payload length is limited to 65535. Application level headers will make the actual payload shorter. Even though binary encodings such as GPB and CBOR may not require more space than what

is left, more voluminous encodings such as JSON and XML may suffer from this size limitation. Although IPv4 and IPv6 senders can fragment outgoing packets exceeding their Maximum Transmission Unit(MTU), fragmented IP packets may not be desired for operational and performance reasons.

Consequently, implementations of the mechanism SHOULD provide a configurable max-fragment-size option to control the maximum size of a payload.

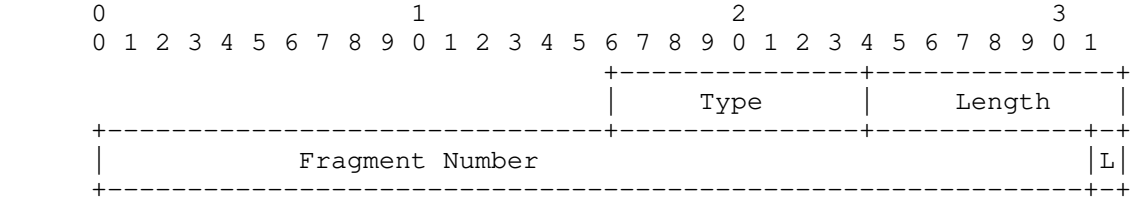


Figure 4: Fragmentation Option Format

The Fragmentation Option is to be included when the message content is fragmented into multiple pieces. Different fragments of one message share the same Message ID. An illustration is provided in Figure 4. The fields of this TLV are:

- Type: indicates Fragmentation Option. The Type value is to be asigned.
- Length: is a fixed value of 6 octets.
- Fragment Number: indicates the sequence number of the current fragment.
- L: is a flag to indicate whether the current fragment is the last one. When 0 is set, the current fragment is not the last one, hence more fragments are expected. When 1 is set, the current fragment is the last one.

3.4. Data Encoding

UDP-Notif message data can be encoded in GPB, CBOR, XML or JSON format. It is conceivable that additional encodings may be supported in the future. This can be accomplished by augmenting the subscription data model with additional identity statements used to refer to requested encodings.

Implementation MAY support multiple encoding methods per subscription. When bundled notifications are supported between the

publisher and the receiver, only subscribed notifications with the same encoding can be bundled in a given message.

4. Congestion Control

Congestion control mechanisms that respond to congestion by reducing traffic rates and establish a degree of fairness between flows that share the same path are vital to the stable operation of the Internet [RFC2914]. While efficient, UDP has no built-in congestion control mechanism. Because streaming telemetry can generate unlimited amounts of data, transferring this data over UDP may be considered problematic. It is not recommended to use the proposed mechanism over congestion-sensitive network paths. The only environments where UDP-Notif is expected to be used are managed networks. The deployments require that the network path has been explicitly provisioned to handle the traffic through traffic engineering mechanisms, such as rate limiting or capacity reservations. The UDP-Notif Message ID can be used to deduce congestion based on packet loss detection. Hence the collector can notify the device to use a lower streaming rate. The interaction to control the streaming rate on the device is out of the scope of this document.

5. Applicability

The target application for UDP-Notif is the collection of data-plane information. The lack of reliability of the data streaming mechanism is thus considered acceptable as the mechanism is to be used in controlled environments, mitigating the risk of information loss, while allowing for publication of very large amounts of data. Moreover, in this context, sporadic events when incomplete data collection is provided is not critical for the proper management of the network.

6. A YANG Data Model for Management of UDP-Notif

The YANG model defined in Section 9 has two leafs augmented into one place of Sub-Notif [RFC8639], plus one identity.

```
 module: ietf-udp-subscribed-notifications
  augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
    +--rw address   inet:ip-address
    +--rw port      inet:port-number
    +--rw enable-fragment?  boolean
    +--rw max-fragment-size?  uint32
```

7. YANG Module

```
<CODE BEGINS> file "ietf-udp-notif@2020-04-27.yang"
module ietf-udp-notif {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-udp-notif";
  prefix un;
  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: Subscription to YANG Notifications";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <http:/tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Authors:  Guangying Zheng
               <mailto:zhengguangying@huawei.com>
               Tianran Zhou
               <mailto:zhoutianran@huawei.com>
               Thomas Graf
               <mailto:thomas.graf@swisscom.com>
               Pierre Francois
               <mailto:pierre.francois@insa-lyon.fr>
               Paolo Lucente
               <mailto:paolo@ntt.net>";

  description
    "Defines UDP-Notif as a supported transport for subscribed
    event notifications.

    Copyright (c) 2018 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 XXXX; see the RFC

    itself for full legal notices.";

  revision 2020-04-27 {
    description
      "Initial version";
    reference
      "RFC XXXX: UDP-based Notifications for Streaming Telemetry";
  }

  identity udp-notif {
    base sn:transport;
    description
      "UDP-Notif is used as transport for notification messages
and state change notifications.";
  }

  identity encode-cbor {
    base sn:encoding;
    description
      "Encode data using CBOR as described in RFC 7049.";
    reference
      "RFC 7049: Concise Binary Object Representation";
  }

  identity encode-gpb {
    base sn:encoding;
    description
      "Encode data using GPB.";
  }

  grouping target-receiver {
    description
      "Provides a reusable description of a UDP-Notif target receiver.";
    leaf address {
      type inet:ip-address;
      mandatory true;
      description
        "IP address of target UDP-Notif receiver, which can be an
        IPv4 address or an IPV6 address.";
    }
    leaf port {
      type inet:port-number;
      mandatory true;
      description
        "Port number of target UDP-Notif receiver, if not specified,
        the system should use default port number.";
```

```
    }

    leaf enable-fragment {
      type boolean;
      default false;
      description
        "The switch for the fragment feature. When disabled, the
         publisher will not allow fragment for a very large data";
    }

    leaf max-fragment-size {
      when "../enable-fragment = true";
      type uint32;
      description "UDP-Notif provides a configurable max-fragment-size
      to control the size of each message.";
    }
  }

  augment "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
    description
      "This augmentation allows UDP-Notif specific parameters to be
       exposed for a subscription.";
    uses target-receiver;
  }
}
<CODE ENDS>
```

8. IANA Considerations

This RFC requests that IANA assigns one UDP port number in the "Registered Port Numbers" range with the service name "udp-notif". This port will be the default port for the UDP-based notification Streaming Telemetry (UDP-Notif) for NETCONF and RESTCONF. Below is the registration template following the rules of [RFC6335].

Service Name: udp-notif

Transport Protocol(s): UDP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: UDP-based Publication Streaming Telemetry

Reference: RFC XXXX

Port Number: PORT-X

IANA is requested to assign a new URI from the IETF XML Registry [RFC3688]. The following URI is suggested:

URI: urn:ietf:params:xml:ns:yang:ietf-udp-notif
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document also requests a new YANG module name in the YANG Module Names registry [RFC7950] with the following suggestion:

name: ietf-udp-notif
namespace: urn:ietf:params:xml:ns:yang:ietf-udp-notif
prefix: un
reference: RFC XXXX

9. Security Considerations

TBD

10. Acknowledgements

The authors of this documents would like to thank Alexander Clemm, Eric Voit, Huiyang Yang, Kent Watsen, Mahesh Jethanandani, Stephane Frenot, Timothy Carey, Tim Jenkins, and Yunan Gu for their constructive suggestions for improving this document.

11. References

11.1. 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>.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <https://www.rfc-editor.org/info/rfc2914>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, DOI 10.17487/RFC4347, April 2006, <https://www.rfc-editor.org/info/rfc4347>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8640] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Dynamic Subscription to YANG Events and Datastores over NETCONF", RFC 8640, DOI 10.17487/RFC8640, September 2019, <https://www.rfc-editor.org/info/rfc8640>.

[RFC8650] Voit, E., Rahman, R., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Dynamic Subscription to YANG Events and Datastores over RESTCONF", RFC 8650, DOI 10.17487/RFC8650, November 2019, <https://www.rfc-editor.org/info/rfc8650>.

11.2. Informative References

[I-D.ietf-netconf-https-notif]
Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for Configured Subscriptions", draft-ietf-netconf-https-notif-02 (work in progress), March 2020.

[I-D.ietf-netconf-notification-messages]
Voit, E., Jenkins, T., Birkholz, H., Bierman, A., and A. Clemm, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-08 (work in progress), November 2019.

[I-D.unyte-netconf-distributed-notif]
Zhou, T., Zheng, G., Voit, E., Graf, T., and P. Francois, "Subscription to Distributed Notifications", draft-unyte-netconf-distributed-notif-00 (work in progress), June 2020.

11.3. URIs

[1] https://developers.google.com/protocol-buffers/

Authors' Addresses

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Thomas Graf
Swisscom
Binzring 17
Zuerich 8045
Switzerland

Email: thomas.graf@swisscom.com

Pierre Francois
INSA-Lyon
Lyon
France

Email: pierre.francois@insa-lyon.fr

Paolo Lucente
NTT
Siriusdreef 70-72
Hoofddorp, WT 2132
NL

Email: paolo@ntt.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 17, 2021

Q. Wu
W. Song
Huawei
L. Geng
P. Liu
China Mobile
Q. Ma
Huawei

October 14, 2020

Adaptive Subscription to YANG Notification
draft-wang-netconf-adaptive-subscription-02

Abstract

This document defines a YANG data model and associated mechanism enabling subscriber's adaptive subscriptions to a publisher's event streams with various different period intervals to report updates. Applying these elements allows both subscriber and publisher to automatically adjust the volume of telemetry traffic sent from publisher to the receivers.

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 April 17, 2021.

Table of Contents

1. Introduction

YANG-Push subscriptions [RFC8641] allow client applications to subscribe to continuous datastore updates without needing to poll. It defines a mechanism (i.e.,update trigger) to determine when an update record needs to be generated. Two type of subscriptions are introduced in [RFC8641], distinguished by how updates are triggered: periodic and on-change.

- Periodic subscription allows subscribed data to be streamed to the destination at a configured fixed periodic interval

- On-change subscription allows update to be triggered whenever a change in the subscribed information is detected. The periodic interval is set to zero value in the on-change subscription case.

However in some large scale deployments (e.g., wireless network performance monitoring) where an increased data collection rate is being used, it becomes more likely that a burst of streamed data may temporarily overwhelm a receiver and consume expensive network resource (e.g., air interface resource). If the rate at which we can collect a stream of data is set too low, these telemetry data are not sufficient to detect and diagnose problems and verify correct network behavior. There is a need for a service to configure both collectors and publishers with multiple different period intervals and automatically switch to different period intervals according to resource usage change, e.g., when the wireless signal strength falls below a configured low watermark, the subscribed data can be streamed at a higher rate while when the wireless signal strength crosses a configured high watermark, the subscribed data can be streamed at lower rate.

This document defines a YANG data model and associated mechanism enabling subscriber's adaptive subscriptions to a publisher's event streams. Applying these elements allows both subscriber and publisher to automatically adjust the volume of telemetry traffic sent from publisher to the receivers.

1.1. Terminology

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.

This document uses the following terms:

Event: An event is something that happens that may be of interest - a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system, for example. Often, this results in an asynchronous message, sometimes referred to as a notification or event notification, being sent to interested parties to notify them that this event has occurred [RFC5277].

Client: Defined in [RFC8342].

Configuration: Defined in [RFC8342].

Configured subscription: Defined in [RFC8639]

Configuration datastore: Defined in [RFC8342].

Notification message: Information intended for a receiver indicating that one or more events have occurred [RFC8639].

Publisher: An entity responsible for streaming notification messages per the terms of a subscription [RFC8639].

Receiver: A target to which a publisher pushes subscribed event records. For dynamic subscriptions, the receiver and subscriber are the same entity [RFC8639].

Subscriber: A client able to request and negotiate a contract for the generation and push of event records from a publisher. For dynamic subscriptions, the receiver and subscriber are the same entity [RFC8639].

Subscription: A contract with a publisher, stipulating the information that one or more receivers wish to have pushed from the publisher without the need for further solicitation [RFC8639].

On-change subscription: A datastore subscription with updates that are triggered when changes in subscribed datastore nodes are detected.

Periodic subscription: A datastore subscription with updates that are triggered periodically according to some time interval.

2. Model Overview

This document defines a YANG module "ietf-adaptive-subscription", which augments the "update-trigger" choice defined in the "ietf-yang-push" module [RFC8641] with subscription configuration parameters that are specific to adaptive subscription.

In addition to Subscription state notifications defined in [RFC8639] and Notifications for Subscribed Content defined in [RFC8641], "ietf-adaptive-subscription" YANG module also defines "adaptive-update" notification to report update interval change.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-adaptive-subscription.yang" module.

```
module: ietf-adaptive-subscription
  augment /sn:subscriptions/sn:subscription/yp:update-trigger:
    +--rw (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
              +--rw adaptive-period* [name]
                 +--rw name                      string
                 +--rw xpath-external-eval       string
                 +--rw watermark?                uint32
                 +--rw period                    centiseconds
                 +--rw anchor-time?              yang:date-and-time
  augment /sn:establish-subscription/sn:input/yp:update-trigger:
    +-- (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
             +--rw adaptive-period* [name]
                +--rw name                      string
                +--rw xpath-external-eval       string
                +--rw watermark?                uint32
                +--rw period                    centiseconds
                +--rw anchor-time?              yang:date-and-time
  notifications:
    +---n adaptive-period-update
       +--ro id?                                      sn:subscription-id
       +--ro period                                   centiseconds
       +--ro anchor-time?                             yang:date-and-time
       +--ro (selection-filter)?
          +--:(by-reference)
          |  +--ro selection-filter-ref               selection-filter-ref
          +--:(within-subscription)
             +--ro (filter-spec)?
                +--:(datastore-subtree-filter)
                |  +--ro datastore-subtree-filter?   <anydata> {sn:subtree}?
                +--:(datastore-xpath-filter)
                   +--ro datastore-xpath-filter?     yang:xpath1.0 {sn:xpath}?
```

2.1. Subscription Configuration

For adaptive subscriptions, triggered updates will occur at the boundaries of specified time intervals when a trigger condition is satisfied. These boundaries can be calculated from the adaptive periodic parameters:

- a "period" that defines the new duration between push updates, the period can be changed based on trigger condition.

- an "anchor-time" update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an

"anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

- a "watermark" that defines the threshold value of the targeted data object, e.g., it can be lower boundary or upper boundary of targeted data object.

- a "xpath-external-eval" represents an Evaluation criteria that may be applied against event records in an event stream, which is used to trigger update interval switch. It contains comparisons of datastore node with specific threshold (i.e., watermark) and associated logical operations in the XPath format. Different from stream-xpath-filter defined in [RFC8639], it doesn't influence the event records output generation from a publisher.

2.2. YANG RPC

2.2.1. "establish-subscription" RPC

The augmentation of YANG module ietf-yang-push made to RPCs specified in YANG module ietf-subscribed-notifications [RFC8639] is introduced. This augmentation concerns the "establish- subscription" RPC, which is augmented with parameters that are needed to specify adaptive subscriptions. These parameters are same as one defined in Section 2.1.

2.2.2. "modify-subscription" RPC

The subscriber MAY invoke the "modify-subscription" RPC for a subscription it previously established. The subscriber will include newly desired values in the "modify-subscription" RPC. Parameters not included MUST remain unmodified. Section 4.4.2 of [RFC8641] provides an example where a subscriber attempts to modify the period and datastore XPath filter of a subscription using NETCONF. The period can be the 'period' parameter defined by ietf-adaptive-subscription.

2.3. Notifications for Adaptive Subscribed Content

The adaptive update notification is similar to Subscription state change notifications defined in [RFC8639]. It is inserted into the sequence of notification messages sent to a particular receiver. The adaptive update notification cannot be dropped or filtered out, it cannot be stored in replay buffers, and it is delivered only to impacted receivers of a subscription. The identification of adaptive update notification is easy to separate from other notification messages through the use of the YANG extension "subscription-state-

notif". This extension tags a notification as a subscription state change notification.

The objects in the 'adpative-update' notification include:

- a "period" that defines the duration between push updates, the period can be changed based on trigger condition.

- an "anchor-time"; update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

- A selection filter identifying YANG nodes of interest in a datastore. Filter contents are specified via a reference to an existing filter or via an in-line definition for only that subscription. Referenced filters allow an implementation to avoid evaluating filter acceptability during a dynamic subscription request. The "case" statement differentiates the options. Note that filter contents are not affected by "xpath-external-eval" parameter and "watermark" parameter defined by update trigger.

3. Adaptive Subscription YANG Module

```
<CODE BEGINS> file "ietf-adaptive-subscription@2020-02-14.yang"
module ietf-adaptive-subscription {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription";
  prefix as;

  import ietf-subscribed-notifications {
    prefix sn;
  }
  import ietf-yang-push {
    prefix yp;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "";
  description
    "NETCONF Protocol Data Types and Protocol Operations.
     Copyright (c) 2020 IETF Trust and the persons identified as
     the document authors.  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
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC xxxx; see
     the RFC itself for full legal notices.";

  revision 2019-12-15 {
    description
      "Initial revision";
    reference
      "RFCxxx Adaptive subscription to YANG notification.";
  }

  typedef centiseconds {
    type uint32;
    description
      "A period of time, measured in units of 0.01 seconds.";
  }

  typedef seconds {
    type uint32;
    description
      "A period of time, measured in units of 1 seconds.";
  }

  typedef operator {
    type enumeration {
      enum unequal {
        description
          "Indicates that the comparision type is unequal to.";
      }
      enum equal {
        description
          "Indicates that the comparision type is equal to.";
      }
      enum less {
        description
          "Indicates that the comparision type is less than.";
      }
      enum less-or-equal {
        description
          "Indicates that the comparision type is less than
           or equal to.";
      }
```

```
      enum greater {
        description
          "Indicates that the comparision type is greater than.";
      }
      enum greater-or-equal {
        description
          "Indicates that the comparision type is greater than
           or equal to.";
      }
    }
    description
      "definition of the operator";
  }

  grouping adaptive-subscription-modifiable {
    description
      "This grouping describes the datastore-specific adaptive subscription
       conditions that can be changed during the lifetime of the
       subscription.";
    choice adaptive-subscription {
      description
        "Defines necessary conditions for sending an event record to
               the subscriber.";
      container adaptive-subscriptions {
        list adaptive-period {
          description
            "Defines necessary conditions to switch update interval for
            sending an event record to the subscriber. The event record output
            generation will not be influeced these conditions.";
          key "name";
          leaf name {
            type string {
          length "1..64";
           }
        description
          "The name of the condition to be matched.  A device MAY further
           restrict the length of this name; space and special
           characters are not allowed.";
         }
          leaf xpath-external-eval {
            type string;
            description
              "A XPath string, representing a logical expression,
               which can contain comparisons of datastore values
               and logical operations in the XPath format.";
          }
          leaf watermark {
            type uint32;
```

```
            description
              "The watermark for targeted data object. The high
               watermark, lowe watermark can be specified for the
               targeted data object.";
          }
          leaf period {
            type centiseconds;
            mandatory true;
            description
              "Duration of time that should occur between periodic
               push updates, in units of 0.01 seconds.";
          }
          leaf anchor-time {
            type yang:date-and-time;
            description
              "Designates a timestamp before or after which a series
               of periodic push updates are determined.  The next
               update will take place at a point in time that is a
               multiple of a period from the 'anchor-time'.
               For example, for an 'anchor-time' that is set for the
               top of a particular minute and a period interval of a
               minute, updates will be sent at the top of every
               minute that this subscription is active.";
          }
        }
        description
          "Container for adaptive subscription.";
      }
    }
  }

  augment "/sn:subscriptions/sn:subscription/yp:update-trigger" {
    description
      "This augmentation adds additional subscription parameters
       that apply specifically to adaptive subscription.";
    uses adaptive-subscription-modifiable;
  }
  augment "/sn:establish-subscription/sn:input/yp:update-trigger" {
    description
      "This augmentation adds additional subscription parameters
         that apply specifically to datastore updates to RPC input.";
    uses adaptive-subscription-modifiable;
  }

  notification adaptive-period-update {
    sn:subscription-state-notification;
    description
      "This notification contains a push update that in turn contains
```

```
       data subscribed to via a subscription.  In the case of a
       periodic subscription, this notification is sent for periodic
       updates.  It can also be used for synchronization updates of
       an on-change subscription.  This notification shall only be
       sent to receivers of a subscription.  It does not constitute
       a general-purpose notification that would be subscribable as
       part of the NETCONF event stream by any receiver.";
    leaf id {
      type sn:subscription-id;
      description
        "This references the subscription that drove the
         notification to be sent.";
    }
    leaf period {
      type centiseconds;
      mandatory true;
      description
        "New duration of time that should occur between periodic
         push updates, in units of 0.01 seconds.";
    }
    leaf anchor-time {
      type yang:date-and-time;
      description
        "Designates a timestamp before or after which a series
         of periodic push updates are determined.  The next
         update will take place at a point in time that is a
         multiple of a period from the 'anchor-time'.
         For example, for an 'anchor-time' that is set for the
         top of a particular minute and a period interval of a
         minute, updates will be sent at the top of every
         minute that this subscription is active.";
    }
    uses yp:datastore-criteria {
      refine "selection-filter/within-subscription" {
        description
          "Specifies the selection filter and where it originated
           from.  If the 'selection-filter-ref' is populated, the
           filter in the subscription came from the 'filters'
           container.  Otherwise, it is populated in-line as part
           of the subscription itself.";
      }
    }
  }
}
<CODE ENDS>
```

4. IANA Considerations

4.1. Updates to the IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

```
---------------------------------------------------------------------
   URI: urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------
```

4.2. Updates to the YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC7950]. . Following the format in [RFC6020], the following registration has been made:

```
---------------------------------------------------------------------
   Name:         ietf-adaptive-subscription
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
   Prefix:       as
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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 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:

- /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:watermark
- /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:period
- /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:anchor-time

6. Contributors

The authors would like to thank Michale Wang for his major contributions to the initial modeling and use cases.

7. References

7.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

7.2. 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>.

[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>.

Appendix A. Example YANG Module

The example YANG module used in this document represents a simple wifi mac interface.

YANG tree diagram for the "example-wifi-mac" module:

```
   module: example-wifi-mac
     +--rw clients
        +--ro client* [mac]
           +--ro mac                yang:mac-address
           +--ro rssi?              int8
           +--ro snr?               uint8
           +--ro ss?                uint8
           +--ro phy-rate?          uint16
           +--ro channel-support*   uint8
           +--ro neighbors
           |  +--ro neighbor-bssid?          yang:mac-address
           |  +--ro neighbor-channel?        uint8
           |  +--ro neighbor-rssi?           int8
           |  +--ro neighbor-antenna?        uint8
           |  +--ro channel-load-report?     uint8
           +--ro ssid
              +--ro name?                    string
              +--ro enabled?                 boolean
              +--ro broadcast-filter?        boolean
              +--ro multicast-filter?        boolean
              +--ro ipv6-ndp-filter?         boolean
              +--ro ipv6-ndp-filter-timer?   uint16
              +--ro station-isolation?       boolean
```

A.1. "example-wifi-mac" YANG Module

```
module example-wifi-mac {
  yang-version 1;
  namespace "http://example.com/yang/wifi-mac";
  prefix wifi;

  import ietf-yang-types {
    prefix yang;
  }

  container clients {
    description
      "Top-level container for clients operational state data.";
    list client {
      key "mac";
      config false;
      description
        "List of clients per BSS.";
      leaf mac {
        type yang:mac-address;
        description
          "MAC address of the client.";
      }
```

```
      leaf rssi {
        type int8;
        description
          "The RSSI of this client in dBm. Expressed as negative
           number";
      }
      leaf snr {
        type uint8;
        description
          "The SNR of AP to Client, in dB.";
      }
      leaf ss {
        type uint8;
        description
          "Number of Spatial Streams supported by the client.";
      }
      leaf phy-rate {
        type uint16;
        description
          "Last used PHY rate of connected client.";
      }
      leaf-list channel-support {
        type uint8;
        description
          "List of supported channels.";
      }
      container neighbors {
        description
          "Container for Client beacon reports. Requires 802.11k
           enabled. See Sec. 5.2.7.1 of 802.11k-2008 Standard.";
        leaf neighbor-bssid {
          type yang:mac-address;
          description
            "The BSSID of this neighbor.";
        }
        leaf neighbor-channel {
          type uint8;
          description
            "The channel of this neighbor.";
        }
        leaf neighbor-rssi {
          type int8;
          description
            "The RSSI of this neighbor in dBm, expressed as a negative
             number.";
        }
        leaf neighbor-antenna {
          type uint8;
```

```
          description
            "Antenna details for this neighbor.";
        }
        leaf channel-load-report {
          type uint8;
          description
            "Channel load, as reported by Client to AP
             normalized to 255. See Sec. 10.11.9.3 of 802.11ac-2013
             Spec.";
        }
      }
      container ssid {
        description
          "Top level container for ssids, including configuration
           and state data.";
        leaf name {
          type string;
          description
            "The name of the SSID.";
        }
        leaf enabled {
          type boolean;
          default "true";
          description
            "The desired operational state (up/down) of this SSID.";
        }
        leaf broadcast-filter {
          type boolean;
          description
            "Convert all downstream broadcast ARP to unicast
             only if Station is associated to the AP. Drop packet
             if Station is not associated to the AP. All other
             broadcast, except DHCP, is dropped by the AP.

             DHCP Offers/ACKs are converted to Unicast, over-the-air.";
        }
        leaf multicast-filter {
          type boolean;
          description
            "Drop all downstream Multicast packets.";
        }
        leaf ipv6-ndp-filter {
          type boolean;
          description
            "Neighbor Advertisements will be cached at the AP (or WLC)
             and unicast in response to Neighbor Solicitations.

             Router Advertisements, in response to a Router Solicitation
```

```
            are converted to Unicast for over-the-air transmission.";
        }
        leaf ipv6-ndp-filter-timer {
          type uint16;
          units "seconds";
          description
            "Time, in seconds, the ndp-filter will cache
             Neighbor Advertisements (NA).";
        }
        leaf station-isolation {
          type boolean;
          description
            "Block Station peer to peer communication.";
        }
      }
    }
  }
}
```

Appendix B. Adaptive Subscription and Notification Example

The examples within this document use the normative YANG module "ietf-adaptive-subscription" as defined in Section 3 and the non-normative example YANG module "example-wifi-mac" as defined in Appendix A.1.

This section shows some typical adaptive subscription and notification message exchanges.

B.1. "edit-config" Example

The client configure adaptive subscription parameters on the server. The adaptive subscription configuration parameters require the server to scan all clients every 5 seconds if the ssid value of client is greater than -65dB; If the ssid value of client is less than -65dB, switch to 60 seconds period value, and then scan all clients every 60 seconds.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <top xmlns="http://example.com/schema/1.2/config">
      <yp:datastore
        xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
        ds:running
      </yp:datastore>
     <yp:datastore-xpath-filter
       xmlns:ex="https://example.com/sample-data/1.0">
       /ex:example-wifi-mac
    </yp:datastore-xpath-filter>
  <as:adaptive-subscriptions
 xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
  <as:adaptive-period>
   <as:condition-expression>as:clients/as:client[ssid > -65]</as:
    condition-expression>
   <as:watermark>-65</as:watermark>
   <as:period>5</as:period>
  </as:adaptive-period>
  <as:adaptive-period>
   <as:condition-expression>as:clients/as:client[ssid < -65]</as:
     condition-expressioni>
   <as:watermark>-65</as:watermark>
   <as:period>60</as:period>
  </as:adaptive-period>
 </as:adaptive-subscriptions>
      </top>
    </config>
  </edit-config>
</rpc>
```

B.2. Create Adaptive Subscription Example

The subscriber sends an "establish-subscription" RPC with the parameters listed in to request the creation of a adaptive subscription. The adaptive subscription configuration parameters require the server to scan all clients every 5 seconds if the ssid value of client is greater than -65dB; If the ssid value of client is less than -65dB, switch to 60 seconds period value, and then scan all clients every 60 seconds. (Section 2)

```
<netconf:rpc message-id="101"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
          xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:running
     </yp:datastore>
     <yp:datastore-xpath-filter
         xmlns:ex="https://example.com/sample-data/1.0">
       /ex:example-wifi-mac
     </yp:datastore-xpath-filter>
     <as:adaptive-subscriptions
    xmlns="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
       <as:adaptive-period>
        <as:condition-expression>as:clients/as:client[ssid > -65]
         </as:condition-expressioni>
        <as:watermark>-65</as:watermark>
        <as:period>5</as:period>
       </as:adaptive-period>
       <as:adaptive-period>
        <as:condition-expression>as:clients/as:client[ssid < -65]
         </as:condition-expressioni>
        <as:watermark>-65</as:watermark>
        <as:period>60</as:period>
       </as:adaptive-period>
     </as:adaptive-subscriptions>
   </establish-subscription>
 </netconf:rpc>
```

In another example, the adaptive subscription configuration parameters could also require the server to scan all clients every 5 seconds if the difference between maximum value of client ssid and minimum value of client ssid is greater than 0.20dB; If the difference between maximum value of client ssid and minimum value of client ssid is less than 20dB, switch to 60 seconds period value and then scan all clients every 60 seconds.

```
<netconf:rpc message-id="101"
     xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <establish-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
          xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:running
     </yp:datastore>
     <yp:datastore-xpath-filter
         xmlns:ex="https://example.com/sample-data/1.0">
       /ex:example-wifi-mac
     </yp:datastore-xpath-filter>
     <as:adaptive-subscriptions>
        <as:data-path>as:clients/as:client</as:data-path>
        <as:target>ssid</as:target>
       <as:adaptive-period>
        <as:condition-expression>as:clients/as:client[max(ssid)-min(ssid) >20]
         </as:condition-expressioni>
        <as:watermark>20</as:watermark>
        <as:period>5</as:period>
       </as:adaptive-period>
       <as:adaptive-period>
        <as:condition-expression>as:clients/as:client[max(ssid)-min(ssid) < 20]
         </as:condition-expressioni>
        <as:watermark>20</as:watermark>
        <as:period>60</as:period>
       </as:adaptive-period>
     </as:adaptive-subscriptions>
   </establish-subscription>
 </netconf:rpc>
```

B.3. "adaptive-update" notification example

Upon the server switches to from the update interval 5 seconds to the new update interval 60 seconds, Before sending event records to receivers, the "adaptive-update" notification should be generated and sent to the receivers to inform the receivers that the update interval value is switched to the new value.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-11-21T13:51:00Z</eventTime>
  <adaptive-update xmlns="http://example.com/ietf-adaptive-subscription">
    <id>0</id>
    <period>60</period>
     <yp:datastore
          xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:running
     </yp:datastore>
     <yp:datastore-xpath-filter
         xmlns:ex="https://example.com/sample-data/1.0">
       /ex:example-wifi-mac
     </yp:datastore-xpath-filter>
  </adaptive-update>
</notification>
```

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Wei Song
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: songwei80@huawei.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: maqiufang1@huawei.com

NETCONF Working Group
Internet-Draft
Intended status: Experimental
Expires: 3 November 2022

Q. Wu
W. Song
Huawei
P. Liu
China Mobile
Q. Ma
Huawei
W. Wang
China Telecom
Z. Niu
Microsoft
2 May 2022

Adaptive Subscription to YANG Notification
draft-wang-netconf-adaptive-subscription-10

Abstract

This document defines a YANG data model and associated mechanism enabling the subscriber's adaptive subscriptions to a publisher's event streams with various different period intervals to report updates. Applying these elements allows servers automatically adjust the rate and volume of telemetry traffic sent from a publisher to receivers.

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 3 November 2022.

Table of Contents

1. Introduction

YANG-Push subscriptions [RFC8641] allow subscriber applications to request a continuous customized stream of updates from a YANG datastore without needing to poll. It defines a mechanism (i.e., update trigger) to determine when an update record needs to be generated. Two types of subscriptions are introduced in [RFC8641], distinguished by how updates are triggered: periodic and on-change.

* Periodic subscription allows subscribed data to be streamed to the destination at a configured fixed periodic interval;

* On-change subscription allows update to be triggered whenever a change in the subscribed information is detected.

However in some large scale deployments (e.g., massive data collection for wireless network performance monitoring) where an increased data collection rate is used, it becomes more likely that both clients and servers are temporarily overwhelmed with a burst of streamed data and consumes expensive network resource (e.g., bandwidth resource, radio resource) and computation resource, therefore hard to continuously monitor operational data, especially values that fall outside normal operational ranges. If the rate at which we can collect a stream of data is set too low or chosen to get low priority telemetry data dropped, these telemetry data are not sufficient to detect and diagnose problems and verify correct network behavior.

A client might choose to monitor the operational state and send a request to modify the data collection rate on the server. But how often the client evaluates if the modification of the data collection rate is required highly depends on the current collection rate, collecting a stream of data at a low rate prevents the subscriber from capturing sufficient data for timely decision-making, which may result in service discontinuity. In addition, when tens of thousands of network devices need to be managed, frequent follow-up modification requests are prone to errors.

There is a need for a service to balance between data management cost and real time streaming telemetry. To achieve this, servers can be configured with multiple different period intervals and corresponding subscription update policy which allows servers/publishers automatically switch to different period intervals according to the network condition change without the interaction with the client for policy update instruction, e.g., when the wireless signal strength falls below a configured threshold, the subscribed data can be streamed at a higher rate to capture potentially important data and events (e.g., continuous service degeneration); while when the wireless signal strength crosses a configured threshold, the subscribed data can be streamed at a lower rate.

This document defines a YANG data model and associated mechanism enabling the subscriber’s adaptive subscriptions to a publisher’s event streams. Applying these elements allows servers to automatically adjust the rate and volume of telemetry traffic sent from a publisher to receivers.

1.1. Terminology

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.

The following terms are defined in [RFC5277] [RFC7950] [RFC3198] [RFC8342] [RFC8639] [RFC8641] and are not redefined here:

* Event
* Client
* Configuration
* Configured subscription
* Configuration datastore
* Notification message
* Publisher
* Receiver
* Subscriber
* Subscription
* On-change subscription
* Periodic subscription
* Selection filter

This document defines the following term:

Adaptive Subscription: Apply subscription update policy on the servers and allow servers/publishers automatically switch to different period intervals according to the network condition change without the interaction with the client for update policy instruction.

2. Model Overview

This document defines a YANG module "ietf-adaptive-subscription", which augments the "update-trigger" choice defined in the "ietf-yang-push" module [RFC8641] with subscription configuration parameters that are specific to a subscriber's adaptive subscription.

In addition to subscription state notifications defined in [RFC8639] and notifications for subscribed content defined in [RFC8641], "ietf-adaptive-subscription" YANG module also defines "adaptive-period-update" notification to report the update interval change.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-adaptive-subscription" module.

```
module: ietf-adaptive-subscription
  augment /sn:subscriptions/sn:subscription/yp:update-trigger:
    +--rw (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
              +--rw adaptive-period* [name]
                 +--rw name                      string
                 +--rw xpath-external-eval       string
                 +--rw period                    yp:centiseconds
                 +--rw anchor-time?              yang:date-and-time
  augment /sn:establish-subscription/sn:input/yp:update-trigger:
    +-- (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
             +--rw adaptive-period* [name]
                +--rw name                      string
                +--rw xpath-external-eval       string
                +--rw period                    yp:centiseconds
                +--rw anchor-time?              yang:date-and-time
  notifications:
    +---n adaptive-period-update
       +--ro id?                                      sn:subscription-id
       +--ro period                                   yp:centiseconds
       +--ro anchor-time?                             yang:date-and-time
       +--ro (selection-filter)?
          +--:(by-reference)
          |  +--ro selection-filter-ref               selection-filter-ref
          +--:(within-subscription)
             +--ro (filter-spec)?
                +--:(datastore-subtree-filter)
                |  +--ro datastore-subtree-filter?   <anydata> {sn:subtree}?
                +--:(datastore-xpath-filter)
                   +--ro datastore-xpath-filter?     yang:xpath1.0 {sn:xpath}?
```

2.1. Subscription Configuration

For adaptive subscriptions, triggered updates will occur at the boundaries of specified time intervals when a trigger condition is satisfied. These boundaries can be calculated from the following adaptive periodic parameters:

* a "name" represents the name of each adaptive period;

* a "period" that defines the new duration between push updates. The period can be switched based on trigger conditions;

* an "anchor-time"; update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

* an "xpath-external-eval" represents a standard XPath evaluation expression (See section 6.4 of [RFC7950]) that is applied against the targeted data object, which is used to trigger/control the update interval switching within the server. It follows the rules defined in section 3.4 of [XPATH1.0] and contains comparisons of the targeted datastore node with its value to the specific threshold in the XPath format. Different from selection filter defined in [RFC8641],

  - it is applied against a single targeted object rather than a set of target objects.

  - it monitors a specific data object change and evaluates the trigger condition associated with the targeted object to be true or false using XPATH rules and does not influence the event records output generation from a publisher.

  How often the XPath expression criterion is evaluated is up to the publisher's implementation. With minimal delay, the expression can be evaluated whenever changes to targeted object occur, or at the end of each high-frequency streaming update period. To reduce the frequency of evaluation, the server can choose to check targeted object change at every multiple (e.g.,2 or 3) high-frequency streaming update periods.

  The represented expression defined in "xpath-external-eval" is evaluated in the following XPath context:

- The set of namespace declarations is the set of prefix and namespace pairs for all YANG modules implemented by the server, where the prefix is the YANG module name and the namespace is as defined by the "namespace" statement in the YANG module.

- If the leaf is encoded in XML, all namespace declarations in scope on the "xpath-external-eval" leaf element are added to the set of namespace declarations. If a prefix found in the XML is already present in the set of namespace declarations, the namespace in the XML is used.

- The set of variable bindings is empty.

- The function library is the core function library defined in [XPATH1.0] and the function defined in Section 10 in RFC 7950.

- The context node is the root node.

For the cases where multiple list instances are needed to handle in "xpath-external-eval", XPath abbreviated syntax can be used to identify a particular instance, e.g., to represent a comparison for a leaf in a list entry:

```
/if:interfaces/if:interface[if:name="eth0"]/if:in-errors>1000.
```

The server MUST convert the XPath expression defined in "xpath-external-eval" to a boolean value and internally apply the "boolean" function defined in Section 4.3 in [XPATH1.0] if the evaluated result is not a boolean.

Note that the adaptive subscription may not be supported by every YANG datastore nodes. A publisher MAY decide to simply reject an adaptive subscription with "adaptive-unsupported" (defined in Section 2.2.1.1) if the scope of the subscription contains selected data nodes for which adaptive subscription is not supported.

2.2. YANG RPC

2.2.1. "establish-subscription" RPC

The augmentation of YANG module "ietf-yang-push" made to RPCs specified in YANG module "ietf-subscribed-notifications" [RFC8639] is introduced. This augmentation concerns the "establish-subscription" RPC, which is augmented with parameters that are needed to specify a subscriber's adaptive subscriptions. These parameters are the same as ones defined in Section 2.1.

2.2.1.1. RPC Failures

As specified in [RFC8639] and [RFC8641], RPC error responses from the publisher are used to indicate a rejection of an RPC for any reason. This document introduces three new RPC errors for "establish-subscription" RPC.

establish-subscription
adaptive-unsupported
xpath-evaluation-unsupported
multi-xpath-criteria-conflict

Adaptive-unsupported is used to indicate that the adaptive subscription is not supported for any objects that are selectable by the filter.

Xpath-evaluation-unsupported is used to indicate that a server fails to parse syntax defined in "xpath-external-eval". The failure can be caused by either a syntax error or some XPath 1.0 syntax not supported against the specific object.

Multi-xpath-criteria-conflict is used to indicate that the multiple Xpath evaluation criteria represented by "xpath-external-eval" is evaluated as conflict, i.e., more than one condition expressions are evaluated to "true". Such a conflict may also cause an ongoing adaptive-subscription terminated.

For an example of how above RPC errors can be returned, see the "xpath-evaluation-unsupported" error response illustrated in Appendix B.3.

2.3. Notifications for Adaptive Subscribed Content

The adaptive update notification is similar to subscription state change notifications defined in [RFC8639]. It is inserted into the sequence of notification messages sent to a particular receiver. The adaptive update notification cannot be dropped or filtered out, it cannot be stored in replay buffers, and it is delivered only to impacted receivers of a subscription. The identification of adaptive update notification is easy to separate from other notification messages through the use of the YANG extension "subscription-state-notif". This extension tags a notification as a subscription state change notification.

The objects in the ’adaptive-period-update’ notification include:

* a "period" that defines the duration between push updates, the period can be changed based on trigger conditions.

* an "anchor-time"; update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

* A selection filter is to identify YANG nodes of interest in a datastore. Filter contents are specified via a reference to an existing filter or via an in-line definition for only that subscription based on XPath Evaluation criteria defined in section 6.4 of [RFC7950]. Referenced filters allow an implementation to avoid evaluating filter acceptability during a dynamic subscription request. The "case" statement differentiates the options. Note that filter contents are not affected by "xpath-external-eval" parameter defined by the update trigger.

3. XPath Complexity Evaluation

YANG-Push subscriptions [RFC8641] specify selection filters to identify targeted YANG datastore nodes and/or datastore subtrees for which updates are to be pushed. In addition, it specifies update policies which contain conditions that trigger generation and pushing of new update records. To support a subscriber's adaptive subscription defined in this document, the trigger condition can also use similar selection filters to express a standard XPath Evaluation criterion (section 6.4 of [RFC7950]) against targeted data objects.

Similar to on-change subscriptions, adaptive subscriptions are particularly effective for data that changes infrequently, the following complex design choices need to be cautious, although these designs have already been well supported by the section 3.4 of [XPATH1.0]:

* Support XPath Evaluation criteria against every data object;

* Support more than one target object selection and operation(e.g., addition, subtraction, division and multiplication) in the XPath evaluation criterion;

* Support any type of node set in the XPath evaluation criterion, e.g., string,int64, uint64, and decimal64 types;

* Both objects in the XPath Evaluation criterion to be compared are node-sets;

* Two objects to be compared are in different data types, e.g., one is an integer, the other is a string

As described in section 6.4 of RFC7950, Numbers in XPath 1.0 are IEEE 754 [IEEE754-2008] double-precision floating-point values; some values of int64, uint64, and decimal64 types cannot be exactly represented in XPath expressions.

If two objects to be compared are in different data types, conversion function is needed to convert different data types into numbers.

If both objects in XPath Evaluation criteria to be compared are node-sets, more computation resources are required which add complexity.

To reduce these complexities, the following design principles are recommended:

* XPath Evaluation criteria against a minimal set of data objects in the data model, the minimal set of data objects can be advertised using Notification capabilities model defined in [RFC9196].

* XPath Evaluation criteria only support condition expressions that filter updates based on numbers.

* One object to be compared in the XPath Evaluation criteria MUST be a leaf data node.

* The other object to be compared in the XPath Evaluation criteria MUST be a number data type.

If a server receives an XPath Evaluation criterion with some XPath syntax unsupported against the specific object, an RPC error with "xpath-evaluation-unsupported" should be returned.

4. Adaptive Subscription YANG Module

<CODE BEGINS> file "ietf-adaptive-subscription@2020-02-14.yang"

```
module ietf-adaptive-subscription {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription";
  prefix as;
  import ietf-subscribed-notifications {
    prefix sn;
  }
  import ietf-yang-push {
    prefix yp;
  }
  import ietf-yang-types {
```

```
      prefix yang;
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group";
    contact
      "";
    description
      "NETCONF Protocol Data Types and Protocol Operations.
       Copyright (c) 2020 IETF Trust and the persons identified as
       the document authors.  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
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC xxxx; see
       the RFC itself for full legal notices.";

    revision 2020-02-14 {
      description
        "Initial revision";
      reference
        "RFCxxx Adaptive subscription to YANG notification.";
    }

    identity adaptive-unsupported {
      base sn:establish-subscription-error;
      description
        "Adaptive-subscription is not supported for any objects
         that are selectable by the filter.";
    }

    identity xpath-evaluation-unsupported {
      base sn:establish-subscription-error;
      description
        "Unable to parse the xpath evaluation criteria defined in
        xpath-external-eval because of a syntax error or some
         XPath 1.0 syntax not supported against the specific object.";
    }

    identity multi-xpath-criteria-conflict {
      base sn:establish-subscription-error;
      base sn:subscription-terminated-reason;
      description
```

```
        "Multiple Xpath evaluation criteria represented by
         'xpath-external-eval' is evaluated as conflict, i.e.,
         more than one condition expressions are evaluated to
        'true'.";
   }

     grouping adaptive-subscription-modifiable {
       description
         "This grouping describes the datastore-specific adaptive subscription
          conditions that can be changed during the lifetime of the
          subscription.";
       choice adaptive-subscription {
         description
           "Defines necessary conditions for sending an event record to
                  the subscriber.";
         container adaptive-subscriptions {
           list adaptive-period {
             key "name";
             description
               "Defines necessary conditions to switch update interval for
               sending an event record to the subscriber. The event record output
               generation will not be influenced these conditions.";
             leaf name {
               type string {
             length "1..64";
              }
           description
             "The name of the condition to be matched.  A device MAY further
              restrict the length of this name; space and special
              characters are not allowed.";
            }
             leaf xpath-external-eval {
               type string;
               description
                 "A XPath string, representing a logical expression,
                  which can contain comparisons of datastore values
                  and logical operations in the XPath format.";
             }
             leaf period {
               type yp:centiseconds;
               mandatory true;
               description
                 "Duration of time that should occur between periodic
                  push updates, in units of 0.01 seconds.";
             }
             leaf anchor-time {
               type yang:date-and-time;
               description
```

```
                "Designates a timestamp before or after which a series
                 of periodic push updates are determined.  The next
                 update will take place at a point in time that is a
                 multiple of a period from the 'anchor-time'.
                 For example, for an 'anchor-time' that is set for the
                 top of a particular minute and a period interval of a
                 minute, updates will be sent at the top of every
                 minute that this subscription is active.";
            }
          }
          description
            "Container for adaptive subscription.";
        }
      }
    }

    augment "/sn:subscriptions/sn:subscription/yp:update-trigger" {
      description
        "This augmentation adds additional subscription parameters
         that apply specifically to adaptive subscription.";
      uses adaptive-subscription-modifiable;
    }
    augment "/sn:establish-subscription/sn:input/yp:update-trigger" {
      description
        "This augmentation adds additional subscription parameters
           that apply specifically to datastore updates to RPC input.";
      uses adaptive-subscription-modifiable;
    }

    notification adaptive-period-update {
      sn:subscription-state-notification;
      description
        "This notification contains a push update that in turn contains
         data subscribed to via a subscription.  In the case of a
         periodic subscription, this notification is sent for periodic
         updates.  It can also be used for synchronization updates of
         an on-change subscription.  This notification shall only be
         sent to receivers of a subscription.  It does not constitute
         a general-purpose notification that would be subscribable as
         part of the NETCONF event stream by any receiver.";
      leaf id {
        type sn:subscription-id;
        description
          "This references the subscription that drove the
           notification to be sent.";
      }
      leaf period {
        type yp:centiseconds;
```

```
        mandatory true;
        description
          "New duration of time that should occur between periodic
           push updates, in units of 0.01 seconds.";
      }
      leaf anchor-time {
        type yang:date-and-time;
        description
          "Designates a timestamp before or after which a series
           of periodic push updates are determined.  The next
           update will take place at a point in time that is a
           multiple of a period from the 'anchor-time'.
           For example, for an 'anchor-time' that is set for the
           top of a particular minute and a period interval of a
           minute, updates will be sent at the top of every
           minute that this subscription is active.";
      }
      uses yp:datastore-criteria {
        refine "selection-filter/within-subscription" {
          description
            "Specifies the selection filter and where it originated
             from.  If the 'selection-filter-ref' is populated, the
             filter in the subscription came from the 'filters'
             container.  Otherwise, it is populated in-line as part
             of the subscription itself.";
        }
      }
    }
  }
  <CODE ENDS>
```

5. IANA Considerations

5.1. Updates to the IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

5.2. Updates to the YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC7950]. Following the format in [RFC6020], the following registration is requested to be made:

```
---------------------------------------------------------------------
   Name:         ietf-adaptive-subscription
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
   Prefix:       as
   Reference:    RFC xxxx
---------------------------------------------------------------------
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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 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:

* /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:period

* /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:anchor-time

* /sn:establish-subscription/sn:input/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:period

* /sn:establish-subscription/sn:input/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:anchor-time

7. Contributors

Thanks Michael Wang, Liang Geng for their major contributions to the initial modeling and use cases.

Michael Wang
Email: wangzitao@huawei.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

8. Acknowledges

We would like to thank Rob Wilton, Thomas Graf, Andy Bierman, Michael Richardson, Henk Birkholz for valuable review on this document, special thanks to Thomas and Michael for organizing the discussion on several relevant drafts and reach the common understanding on the concept and ideas. Thanks Michael for providing CHIP/Matter WIFI statistics reference.

9. References

9.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

[RFC9196] Lengyel, B., Clemm, A., and B. Claise, "YANG Modules Describing Capabilities for Systems and Datastore Update Notifications", RFC 9196, DOI 10.17487/RFC9196, February 2022, <https://www.rfc-editor.org/info/rfc9196>.

9.2. Informative References

[CHIP] CSA, "Connected Home over IP Specification", April 2021.

[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>.

[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>.

[XPATH1.0] W3C, "https://www.w3.org/TR/1999/REC-xpath-19991116/", 11 November 1999.

Appendix A. Example YANG Module

The example YANG module used in this document represents a Wi-Fi Network Diagnostics data specified in [CHIP] which can be used by a Node to assist a user or Administrative Node in diagnosing potential problems.

YANG tree diagram for the "example-wifi-network-diagnostic" module:

```
module: example-wifi-network-diagnostic
  +--rw server
  |  +--rw bssid?                        yang:mac-address
  |  +--rw security-type?                enumeration
  |  +--rw wifi-version?                 enumeration
  |  +--rw channel-num?                  int8
  |  +--rw rssi?                         int8
  |  +--rw beacon-lost-count?            int8
  |  +--rw beacon-rx-count?              int8
  |  +--rw packet-multicast-rx-count?    int8
  |  +--rw packet-multicast-tx-count?    int8
  |  +--rw packet-unicast-rx-count?      int8
  |  +--rw packet-unicast-tx-count?      int8
  |  +--rw current-max-rate?             int8
  |  +--rw overrun-count?                int8
  +--rw events
     +--rw event* [name]
        +--rw name                   string
        +--rw disconnection?         enumeration
        +--rw association-failure?   enumeration
        +--rw connection-status?     enumeration
```

A.1. "example-wifi-mac" YANG Module

```
module example-wifi-network-diagnostic {
  yang-version 1;
  namespace "http://example.com/yang/wifi-network-diagnostic";
  prefix wnd;

  import ietf-yang-types {
    prefix yang;
  }

  container server {
    description
      "Configuration of the WiFi Server logical entity.";
    leaf bssid {
      type yang:mac-address;
      description
        "The MAC address of a wireless access point.";
    }
    leaf security-type {
      type enumeration {
        enum unspecified {
          value 0;
        }
        enum none {
          value 1;
        }
        enum wep {
          value 2;
        }
        enum wpa {
          value 3;
        }
        enum wpa2 {
          value 4;
        }
        enum wpa3 {
          value 5;
        }
      }
      description
        "The type of Wi-Fi security used. A value of 0
         indicate that the interface is not currently
         configured or operational.";
    }
    leaf wifi-version {
      type enumeration {
        enum 80211a {
          value 0;
        }
```

```
          enum 80211b {
            value 1;
          }
          enum 80211g {
            value 2;
          }
          enum 80211n {
            value 3;
          }
          enum 80211ac {
            value 4;
          }
          enum 80211ax {
            value 5;
          }
        }
        description
          "The highest 802.11 standard version usable
           by the Node.";
      }
      leaf channel-num {
        type int8;
        description
          "The channel that Wi-Fi communication is currently
           operating on. A value of 0indicates that the interface
           is not currently configured or operational.";
      }
      leaf rssi {
        type int8;
        description
          "The RSSI of the Nodes Wi-Fi radio in dBm.";
      }
      leaf beacon-lost-count {
        type int8;
        description
          "The count of the number of missed beacons the
           Node has detected.";
      }
      leaf beacon-rx-count {
        type int8;
        description
          "The count of the number of received beacons. The
           total number of expected beacons that could have been
           received during the interval since association SHOULD
           match the sum of BeaconRxCount and BeaconLostCount. ";
      }
      leaf packet-multicast-rx-count {
        type int8;
```

```
        description
          "The number of multicast packets received by
           the Node.";
        }
        leaf packet-multicast-tx-count {
          type int8;
          description
            "The number of multicast packets transmitted by
             the Node.";
        }
        leaf packet-unicast-rx-count {
          type int8;
          description
            "The number of multicast packets received by
             the Node.";
        }
        leaf packet-unicast-tx-count {
          type int8;
          description
            "The number of multicast packets transmitted by
             the Node.";
        }
        leaf current-max-rate {
          type int8;
          description
            "The current maximum PHY rate of transfer of
             data in bytes-per-second.";
        }
        leaf overrun-count {
          type int8;
          description
            "The number of packets dropped either at ingress or
             egress, due to lack of buffer memory to retain all
             packets on the ethernet network interface. The
             OverrunCount attribute SHALL be reset to 0 upon a
             reboot of the Node..";
        }
      }
      container events {
        description
          "Configuration of WIFI Network Diagnostic events.";
        list event {
          key "name";
          description
            "The list of event sources configured on the
             server.";
          leaf name {
            type string;
```

```
          description
            "The unique name of an event source.";
        }
        leaf disconnection {
          type enumeration {
            enum de-authenticated {
              value 1;
            }
            enum dis-association {
              value 2;
            }
          }
          description
            "A Nodes Wi-Fi connection has been disconnected as a
             result of de-authenticated or dis-association and
             indicates the reason.";
        }
        leaf association-failure {
          type enumeration {
            enum unknown {
              value 0;
            }
            enum association-failed {
              value 1;
            }
            enum authentication-failed {
              value 2;
            }
            enum ssid-not-found {
              value 3;
            }
          }
          description
            "A Node has attempted to connect, or reconnect, to
             a Wi-Fi access point, but is unable to successfully
             associate or authenticate, after exhausting all
             internal retries of its supplicant.";
        }
        leaf connection-status {
          type enumeration {
            enum connected {
              value 1;
            }
            enum notconnected {
              value 2;
            }
          }
          description
```

```
              "A Node's connection status to a Wi-Fi network has
               changed. Connected, in this context, SHALL mean that
               a Node acting as a Wi-Fi station is successfully
               associated to a Wi-Fi Access Point.";
          }
        }
      }
    }
```

Appendix B. Adaptive Subscription and Notification Example

The examples within this document use the normative YANG module "ietf-adaptive-subscription" defined in Section 4 and the non-normative example YANG module "example-wifi-network-diagnostic" defined in Appendix A.1.

This section shows some typical adaptive subscription and notification message exchanges.

B.1. "edit-config" Example

The client configures adaptive subscription policy parameters on the server. The adaptive subscription configuration parameters require the server to support two update intervals (i.e., 5 seconds, 60 seconds) and report updates every 60 seconds if the rssi value is greater than or equal to -65dB; If the rssi value is less than -65dB, switch to 5 seconds period value to report updates.

```
 <rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
   <target>
    <running/>
   </target>
   <config
    xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <top
     xmlns="http://example.com/schema/1.2/config"
     xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
              ds:running
     </yp:datastore>
     <yp:datastore-xpath-filter
      xmlns:wnd="https://example.com/sample-data/1.0">
             /wnd:example-wifi-network-diagnostic
     </yp:datastore-xpath-filter>
     <as:adaptive-subscriptions
      xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
      <as:adaptive-period>
       <as:xpath-external-eval>
        /wnd:server/wnd:rssi&lt;-65
       </as:xpath-external-eval>
       <as:period>5</as:period>
      </as:adaptive-period>
      <as:adaptive-period>
       <as:xpath-external-eval>
        /wnd:server/wnd:rssi&gt;=-65
       </as:xpath-external-eval>
       <as:period>60</as:period>
     </as:adaptive-period>
     </as:adaptive-subscriptions>
    </top>
   </config>
  </edit-config>
 </rpc>
```

B.2. Create Adaptive Subscription Example

The subscriber sends an "establish-subscription" RPC with the parameters listed in to request the creation of an adaptive subscription. The adaptive subscription configuration parameters require the server to report updates every 5 seconds if the rssi value is less than -65dB; If the rssi value is greater than or equal to -65dB, switch to 60 seconds period value. (Section 2)

```
<netconf:rpc message-id="101"
 xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
 <establish-subscription
  xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
  xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
  <yp:datastore
   xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
          ds:running
  </yp:datastore>
  <yp:datastore-xpath-filter
   xmlns:wnd="https://example.com/sample-data/1.0">
       /wnd:example-wifi-network-diagnostic
  </yp:datastore-xpath-filter>
  <as:adaptive-subscriptions
   xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
   <as:adaptive-period>
    <as:xpath-external-eval>
     wnd:server/wnd:rssi&lt;-65
    </as:xpath-external-eval>
    <as:period>5</as:period>
   </as:adaptive-period>
   <as:adaptive-period>
    <as:xpath-external-eval>
     wnd:server/wnd:rssi&gt;=-65
    </as:xpath-external-eval>
    <as:period>60</as:period>
   </as:adaptive-period>
  </as:adaptive-subscriptions>
 </establish-subscription>
</netconf:rpc>
```

B.3. "xpath-evaluation-unsupported" error response example

If the subscriber has authorization to establish the subscription with a server, but the server had not been able to fully satisfy the request from the subscriber, the server should send an RPC error response.

For instance, if the XPATH 1.0 syntax against the targeted data object defined in "xpath-external-eval" is not supported by the server' s implementation, the server returns a reply indicating a failure. The following <rpc-reply> illustrates an example:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-app-tag>
      ietf-adaptive-subscription:xpath-evaluation-unsupported
    </error-app-tag>
    <error-path xmlns:wnd="https://example.com/sample-data/1.0">
      /wnd:server/wnd:rssi
    </error-path>
  </rpc-error>
</rpc-reply>
```

Since adaptive subscription allows a server to be configured with multiple different period intervals and corresponding XPath evaluation criteria to trigger update interval switch in the server, it may be possible for the server to return multiple <rpc-error> elements with "xpath-evaluation-unsupported" failure specified by different error paths. The subscriber can use this information in future attempts to establish a subscription.

B.4. "adaptive-period-update" notification example

Upon the server switches from the update interval 5 seconds to the new update interval 60 seconds, before sending event records to receivers, the "adaptive-period-update" notification should be generated and sent to the receivers to inform the receivers that the update interval value is switched to the new value.

```
<notification
 xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
 xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
 <eventTime>2016-11-21T13:51:00Z</eventTime>
 <adaptive-period-update
  xmlns="http://example.com/ietf-adaptive-subscription">
  <id>0</id>
  <period>60</period>
  <yp:datastore
   xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
       ds:running
  </yp:datastore>
  <yp:datastore-xpath-filter
   xmlns:ex="https://example.com/sample-data/1.0">
       /ex:example-wifi-network-diagnostic
  </yp:datastore-xpath-filter>
 </adaptive-period-update>
</notification>
```

B.5. Changes between Revisions

v09 -v10

* Change the draft intended status to "experimental"

* Problem statement refinement

v08 -v09

* Define two new RPC errors to report when adaptive subscription unsupported or multiple XPath criteria conflict.

* Remove the "watermark" parameter.

* Add clarification about how to evaluate the XPath expression defined in "xpath-external-eval".

* Add clarification about how to compare a targeted data object in a specific list entry.

v07 -v08

* Define a new RPC error to report when an XPath syntax defined in "xpath-external-eval" is unsupported by a server.

* Add a new example showing how the RPC error being returned by a publisher.

* The usage examples fixed in the Appendix.

* Grammatical errors correction(missing articles, plurality mismatches, etc).

v06 -v07

* The usage examples typo fixed in the Appendix.

* Add reference to RFC7950 XPATH Evaluation section and XPATH 1.0

* Clarify the definitions of 'xpath-external-eval' and 'selection-filter' by reusing XPATH Evaluation rules in RFC7950.

* Add a new terminology "adaptive subscription".

* Add one section to discuss Arbitrary XPath Complexity.

v05 -v06

* Replace example-wifi-mac module with example-wifi-network-diagnostic using WIFI statistics specified in CHIP specification.

* Update adaptive subscription Example to align with WIFI example module change.

* Add one more reference to CHIP Specification.

v04 -v05

* Remove "modify-subscption" RPC usage.

* Module update to fix the nits.

* Update adaptive subscription Example.

* Other Editorial changes.

v03 - v04

* Add missing subtrees and data nodes in the security section;

* Change "adaptive-update" notification into "adaptive-period-update" notification;

* Other Editorial changes.

v02 - v03

* Clarify the difference between low priority telemetry data dropping and collection rate switching in the introduction section;

* Update the abstract and introduction section to focus on collection rate switching in the server without interaction with the remote client;

* Format usage example and change ssid into rssi in the appendix;

* Use boilerplate and reuse the terms in the terminology section.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: bill.wu@huawei.com

Wei Song
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: songwei80@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing
Email: liupengyjy@chinamobile.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

Wei Wang
China Telecom
32 Xuanwumen West St, Xicheng District
Beijing
Email: wangw36@chinatelecom.cn

Zhixiong Niu
Microsoft
Email: Zhixiong.Niu@microsoft.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2021

Z. Wang
Q. Wu
Huawei
P. Liu
H. Cai
China Mobile

July 3, 2020

Bulk Subscription to YANG Event Notification
draft-wang-netconf-bulk-subscribed-notifications-02

Abstract

This document defines a YANG data model and associated mechanism that allows subscriber applications to bulk subscribe to publishers' event streams based on bundle group information such as bundle size and bundle latency. This allows the publishers to report multiple notifications in a single bundling message.

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 January 4, 2021.

Table of Contents

1. Introduction

Subscription to YANG Notifications for Datastore Updates [RFC8641] uses a "target" object in subscription protocol operation for identifying the targeted source of information against which the subscription is applied and supports multiple subscriptions on a single transport session. Notification Message Headers and Bundles [I-D.ietf-netconf-notification-messages] allows multiple notifications bundled into one transportable message. However the subscription protocol operation doesn't provide specific criteria to classify subscriptions and therefore lacks the capability to explicitly indicate which specific subscription associated with the notification should be bundled together, e.g., subscription A and B are bundled based on their relationship with a set of YANG data models while subscription C and D are bundled based on "transport" and "encoding" parameters,both bundled groups are transported in the same transport session. A bundle size, bundle latency associated with a set of subscriptions or YANG data models enables the ability to perform encapsulation operation on a set of subscriptions with common characteristics via a single transaction. The bundle size and bundle latency provides a more optimal mechanism for notification encapsulation which would otherwise require multiple atomic transactions on a per subscription (i.e., one message per notification) basis. Following are some of the use-cases where such identifier can be used.

- For a dynamic subscription, the subscriber may have already had priori knowledge about correlation relation between subscriptions(e.g., aggregated subscribed data from multiple sources). With this priori knowledge,it might send a bundle subscription RPC request to indicate what specific notifications associated with the subscription must be bundled together.

- For a configured subscription, self-explanation data Node tag capability advertisment describing correlation between data nodes in different YANG data model from different publisher can be used to further establish correlation relation between subscriptions. The correlation relation between subscriptions can be configured back onto publisher, which help determine which notifications can be bundles and which notifications are not.

- With the above bundle subscription indication from subscriber to the publisher, multiple notifications corresponding to the request protocol operation for those notifications are bundled into one transportable message using Notification Message Headers and Bundles defined in [I-D.ietf-netconf-notification-messages].

This document defines a YANG data model and associated mechanism that classify subscription based on various different filtering criteria and allow subscriber applications to bulk subscribe/unsubscribe to publisher's targeted object source based on bundle size and bundle latency. This also allows the publishers to report multiple notification in a single bundling message defined in [I-D.ietf-netconf-notification-messages].

1.1. Terminology

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.

This document uses the following terms:

Event: An event is something that happens that may be of interest - a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system, for example. Often, this results in an asynchronous message, sometimes referred to as a notification or event notification, being sent to interested parties to notify them that this event has occurred [RFC5277].

Client: Defined in [RFC8342].

Configuration: Defined in [RFC8342].

Configured subscription: Defined in [RFC8639]

Configuration datastore: Defined in [RFC8342].

Event record: A set of information detailing an event [RFC8639].

Event stream: A continuous, chronologically ordered set of events aggregated under some context [RFC8639].

Notification message: Information intended for a receiver indicating that one or more events have occurred [RFC8639].

Publisher: An entity responsible for streaming notification messages per the terms of a subscription [RFC8639].

Receiver: A target to which a publisher pushes subscribed event records. For dynamic subscriptions, the receiver and subscriber are the same entity [RFC8639].

Subscriber: A client able to request and negotiate a contract for the generation and push of event records from a publisher. For dynamic subscriptions, the receiver and subscriber are the same entity [RFC8639].

Subscription: A contract with a publisher, stipulating the information that one or more receivers wish to have pushed from the publisher without the need for further solicitation [RFC8639].

2. Model Overview

The YANG data model for the Bulk Subscriptions to YANG Event Notification has been split into two modules:

- The ietf-bulk-subscription.yang module defines a list for classifying different subscriptions corresponding to target object into groups. Each group is associated with a bundle size, bundle latency and a set of subscriptions. A set of subscription is identified by a "group-id" string. This string is used both as an index within the bulk subscription module. It associate a specific bundle group with a group of subscriptions and a set of YANG data models, as shown in the subscription augmentation. In addition,ietf-subscribed-notifications.yang module defined in [RFC8639] is augmented with "max-bundle-size","max-bundle-latency" and "compression-algorithm" to enhance QoS feature and provide additional subscription bundle classification criteria.

- The ietf-bulk-notification.yang module augment the YANG structure of ietf-notification-messages.yang [draft-ietf-netconf-notification-messages], a "group-id" is added to the "message-header" of the ietf-notification-messages.yang to indicate the group to which a set of notifications belongs. In addition, "compression-algorithm" parameter is augmented to "message-header" to inform the corresponding recievers of compression algorithm to be used by the publisher.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-bulk-subscription.yang" module and the "ietf-bulk-notification.yang" module.

```
module: ietf-bulk-subscription
  +--rw bundle-groups
     +--rw bundle-group* [group-id]
        +--rw group-id    string
        +--rw subscription-id*     leafref
        +--rw yang-module*         yang:yang-identifier

  augment /sn:subscriptions/sn:subscription:
     +--rw max-bundle-size         uint32
     +--rw max-bundle-latency      uint32
     +--rw compression-algorithm  string

  +---x bundle-subscription
     +---input
        +---w group-id?        -> /bundle-groups/bundle-group/group-id
        +--rw max-bundle-size         uint32
        +---w max-bundle-latency      uint32
        +---w compression-algorithm   string
        +---w subscription-id*        subscription-id
        +---w masked-subscription-id* subscription-id

module: ietf-bulk-notification
     augment-structure /nm:message/nm:message-header:
        +--rw group-id?    string
        +--rw compression-algorithm string
```

3. Bulk Subscription YANG Module

```
<CODE BEGINS> file "ietf-bulk-subscription@2019-09-23.yang"
module ietf-bulk-subscription {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bulk-subscription";
  prefix bs;

  import ietf-subscribed-notifications {
```

```
    prefix sn;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "";
  description
    "NETCONF Protocol Data Types and Protocol Operations.

     Copyright (c) 2011 IETF Trust and the persons identified as
     the document authors.  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
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 6241; see
     the RFC itself for full legal notices.";

  revision 2019-09-23 {
    description
      "Initial revision";
       reference "RFC XXXX: Bulk Subscription to YANG Event Notification";
  }
  identity encode-cbor {
    base sn:encoding;
    description
      "Encode data using cbor.";
  }
  identity encode-gpb {
    base sn:encoding;
    description
      "Encode data using gpb.";
  }
  container bundle-groups {
    list bundle-group {
      key "group-id";
      leaf group-id {
        type string;
        description
          "This group ID is used as an index within the bulk subscription module
```

```
           , which indicates subscription classification criteria.";
      }
      leaf-list subscription-id {
        type leafref {
         path "/sn:subscriptions/sn:subscription/sn:id";
        }
        description
          "subscription-id";
      }
    leaf-list yang-module {
      type yang:yang-identifier;
      description
        "Name of the YANG module list supported by the publisher.";
    }
      description
        "List for group that classify different subscriptions into groups.";
    }
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }
    description
      "Container for subscription group.";
  }
  augment "/sn:subscriptions/sn:subscription" {
      leaf max-bundle-size {
      type uint32;
      default 400;
      description
      "The maximum number of data objects to be included by the publisher in a
       single message to recievers.";
     }
      leaf max-bundle-latency {
      type uint32;
      units centiseconds;
      default 400;
      description
      "The maximum latency before a specific YANG Notifications generated
       must egress a publisher.";
     }
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }
```

```
    description
      "Augment the subscribed-notifications module with transport specific inform
ation.";
  }

  rpc bundle-subscription {
   description
    "Bundle subscription. This paremeter indicates what subscription must be bund
led together.";
    input {
     leaf group-id {
      type string;
      description
      "This group ID is used as an index within the bulk subscription module";
      }
    leaf-list subscription-id {
      type uint32;
      description
       "Subscription-id paremeter indicates what subscription must be bundled tog
ether.";
      }
   leaf-list mask-subscription-id {
      type uint32;
      description
       "Mask subscription-id paremeter indicates what subscription must
        not be bundled together.";
      }
      leaf max-bundle-size {
      type uint32;
      default 400;
      description
      "The maximum number of data objects to be included by the publisher in a si
ngle
       message to recievers.";
     }
     leaf max-bundle-latency {
      type uint32;
      units centiseconds;
      default 400;
      description
      "The maximum latency before a specific YANG Notifications generated
       must egress a publisher.";
     }
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }
    }
  }
}
```

<CODE ENDS>

4. Bulk Notification YANG Module

```
<CODE BEGINS> file "ietf-bulk-notification@2019-09-23.yang"
module ietf-bulk-notification {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bulk-notification";
  prefix bn;

  import ietf-yang-structure-ext {
    prefix sx;
  }
  import ietf-notification-messages {
    prefix nm;
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "";
  description
    "NETCONF Protocol Data Types and Protocol Operations.

     Copyright (c) 2011 IETF Trust and the persons identified as
     the document authors.  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
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 6241; see
     the RFC itself for full legal notices.";

  revision 2019-09-23 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Bulk Subscription to YANG Event Notification";
  }

  sx:augment-structure "/nm:message/nm:message-header" {
     leaf group-id {
      type string;
      description
```

```
        "To identify the group to which a set of notifications belongs.";
    }
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }
    description
      "Group related informations are added to the 'message-header' of
       the ietf-notification-messages to identify the group to which a
       set of notifications belongs and compression algorithms used by
       the publisher.";
  }
}
<CODE ENDS>
```

5. IANA Considerations

5.1. Updates to the IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-bulk-subscription
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-bulk-notification
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

5.2. Updates to the YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC7950]. . Following the format in [RFC6020], the following registration has been made:

```
------------------------------------------------------------------
   Name:         ietf-bulk-subscription
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-bulk-subscription
   Prefix:       trig
   Reference:    RFC xxxx

   Name:         ietf-bulk-notification
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-bulk-notification
   Prefix:       evt
   Reference:    RFC xxxx
------------------------------------------------------------------
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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 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:

- /bundle-groups/bundle-group/group-id
- /sn:subscriptions/sn:subscription/bs:max-bundle-latency
- /sn:subscriptions/sn:subscription/bs:max-bundle-size
- /bundle-subscription/input/group-id

7. Acknowledgements

Thanks to Eric Voit for reviewing this draft and providing important input to this document.

8. References

8.1. 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>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

8.2. 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>.

[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>.

Authors' Addresses

Michael Wang
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: wangzitao@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

Hui Cai
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: caihui@chinamobile.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2021

Q. Wu
Huawei
P. Liu
Y. Fu
China Mobile

October 29, 2020

Bulk Subscription to YANG Event Notification
draft-wang-netconf-bulk-subscribed-notifications-03

Abstract

This document defines a YANG data model and associated mechanism that allows subscriber applications to bulk subscribe to publishers' event streams based on bundle group information such as bundle size and bundle latency. This allows the publishers to report multiple notifications in a single bundling message.

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 May 2, 2021.

Table of Contents

1. Introduction

Subscription to YANG Notifications for Datastore Updates [RFC8641] uses a "target" object in subscription protocol operation for identifying the targeted source of information against which the subscription is applied and supports multiple subscriptions on a single transport session. Notification Message Headers and Bundles [I-D.ietf-netconf-notification-messages] allows multiple notifications bundled into one transportable message. However the subscription protocol operation doesn't provide specific criteria to classify subscriptions and therefore lacks the capability to explicitly indicate which specific subscription associated with the notification should be bundled together, e.g., subscription A and B are bundled based on their relationship with a set of YANG data models while subscription C and D are bundled based on "transport" and "encoding" parameters,both bundled groups are transported in the same transport session. A bundle size, bundle latency associated with a set of subscriptions or YANG data models enables the ability to perform encapsulation operation on a set of subscriptions with common characteristics via a single transaction. The bundle size and bundle latency provides a more optimal mechanism for notification encapsulation which would otherwise require multiple atomic transactions on a per subscription (i.e., one message per notification) basis. Following are some of the use-cases where such identifier can be used.

- For a dynamic subscription, the subscriber may have already had priori knowledge about correlation relation between

subscriptions(e.g., aggregated subscribed data from multiple sources). With this priori knowledge,it might send a bundle subscription RPC request to indicate what specific notifications associated with the subscription must be bundled together.

- For a configured subscription, self-explanation data Node tag capability advertisment describing correlation between data nodes in different YANG data model from different publisher can be used to further establish correlation relation between subscriptions. The correlation relation between subscriptions can be configured back onto publisher, which help determine which notifications can be bundles and which notifications are not.

- With the above bundle subscription indication from subscriber to the publisher, multiple notifications corresponding to the request protocol operation for those notifications are bundled into one transportable message using Notification Message Headers and Bundles defined in [I-D.ietf-netconf-notification-messages].

This document defines a YANG data model and associated mechanism that classify subscription based on various different filtering criteria and allow subscriber applications to bulk subscribe/unsubscribe to publisher's targeted object source based on bundle size and bundle latency. This also allows the publishers to report multiple notification in a single bundling message defined in [I-D.ietf-netconf-notification-messages].

1.1. Terminology

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.

This document uses the following terms:

Event: An event is something that happens that may be of interest - a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system, for example. Often, this results in an asynchronous message, sometimes referred to as a notification or event notification, being sent to interested parties to notify them that this event has occurred [RFC5277].

Client: Defined in [RFC8342].

Configuration: Defined in [RFC8342].

Configured subscription: Defined in [RFC8639]

Configuration datastore: Defined in [RFC8342].

Event record: A set of information detailing an event [RFC8639].

Event stream: A continuous, chronologically ordered set of events aggregated under some context [RFC8639].

Notification message: Information intended for a receiver indicating that one or more events have occurred [RFC8639].

Publisher: An entity responsible for streaming notification messages per the terms of a subscription [RFC8639].

Receiver: A target to which a publisher pushes subscribed event records. For dynamic subscriptions, the receiver and subscriber are the same entity [RFC8639].

Subscriber: A client able to request and negotiate a contract for the generation and push of event records from a publisher. For dynamic subscriptions, the receiver and subscriber are the same entity [RFC8639].

Subscription: A contract with a publisher, stipulating the information that one or more receivers wish to have pushed from the publisher without the need for further solicitation [RFC8639].

2. Model Overview

The YANG data model for the Bulk Subscriptions to YANG Event Notification has been split into two modules:

- The ietf-bulk-subscription.yang module defines a list for classifying different subscriptions corresponding to target object into groups. Each group is associated with a bundle size, bundle latency and a set of subscriptions. A set of subscription is identified by a "group-id" string. This string is used both as an index within the bulk subscription module. It associate a specific bundle group with a group of subscriptions and a set of YANG data models, as shown in the subscription augmentation. In addition,ietf-subscribed-notifications.yang module defined in [RFC8639] is augmented with "max-bundle-size","max-bundle-latency" and "compression-algorithm" to enhance QoS feature and provide additional subscription bundle classification criteria.

- The ietf-bulk-notification.yang module augment the YANG structure of ietf-notification-messages.yang [draft-ietf-netconf-

notification-messages], a "group-id" is added to the "message-header" of the ietf-notification-messages.yang to indicate the group to which a set of notifications belongs. In addition, "compression-algorithm" parameter is augmented to "message-header" to inform the corresponding recievers of compression algorithm to be used by the publisher.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-bulk-subscription.yang" module and the "ietf-bulk-notification.yang" module.

```
module: ietf-bulk-subscription
  +--rw bundle-groups
     +--rw bundle-group* [group-id]
        +--rw group-id    string
        +--rw subscription-id*     leafref
        +--rw yang-module*         yang:yang-identifier

  augment /sn:subscriptions/sn:subscription:
     +--rw max-bundle-size         uint32
     +--rw max-bundle-latency      uint32
     +--rw compression-algorithm  string

  +---x bundle-subscription
     +---input
        +---w group-id?        -> /bundle-groups/bundle-group/group-id
        +--rw max-bundle-size         uint32
        +---w max-bundle-latency      uint32
        +---w compression-algorithm   string
        +---w subscription-id*        subscription-id
        +---w masked-subscription-id* subscription-id

module: ietf-bulk-notification
     augment-structure /nm:message/nm:message-header:
        +--rw group-id?    string
        +--rw compression-algorithm string
```

3. Bulk Subscription YANG Module

```
<CODE BEGINS> file "ietf-bulk-subscription@2019-09-23.yang"
module ietf-bulk-subscription {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bulk-subscription";
  prefix bs;

  import ietf-subscribed-notifications {
    prefix sn;
  }
```

```
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "";
  description
    "NETCONF Protocol Data Types and Protocol Operations.

     Copyright (c) 2011 IETF Trust and the persons identified as
     the document authors.  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
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 6241; see
     the RFC itself for full legal notices.";

  revision 2019-09-23 {
    description
      "Initial revision";
       reference "RFC XXXX: Bulk Subscription to YANG Event Notification";
  }
  identity encode-cbor {
    base sn:encoding;
    description
      "Encode data using cbor.";
  }
  identity encode-gpb {
    base sn:encoding;
    description
      "Encode data using gpb.";
  }
  container bundle-groups {
    list bundle-group {
      key "group-id";
      leaf group-id {
        type string;
        description
          "This group ID is used as an index within the bulk subscription module
           , which indicates subscription classification criteria.";
      }
```

```
      leaf-list subscription-id {
        type leafref {
         path "/sn:subscriptions/sn:subscription/sn:id";
        }
        description
          "subscription-id";
      }
    leaf-list yang-module {
      type yang:yang-identifier;
      description
        "Name of the YANG module list supported by the publisher.";
    }
      description
        "List for group that classify different subscriptions into groups.";
    }
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }
    description
      "Container for subscription group.";
  }
  augment "/sn:subscriptions/sn:subscription" {
      leaf max-bundle-size {
      type uint32;
      default 400;
      description
      "The maximum number of data objects to be included by the publisher in a
       single message to recievers.";
     }
      leaf max-bundle-latency {
      type uint32;
      units centiseconds;
      default 400;
      description
      "The maximum latency before a specific YANG Notifications generated
       must egress a publisher.";
     }
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }

    description
```

```
      "Augment the subscribed-notifications module with transport specific inform
ation.";
  }

  rpc bundle-subscription {
   description
    "Bundle subscription. This paremeter indicates what subscription must be bund
led together.";
    input {
     leaf group-id {
      type string;
      description
      "This group ID is used as an index within the bulk subscription module";
      }
    leaf-list subscription-id {
      type uint32;
      description
       "Subscription-id paremeter indicates what subscription must be bundled tog
ether.";
      }
   leaf-list mask-subscription-id {
      type uint32;
      description
       "Mask subscription-id paremeter indicates what subscription must
        not be bundled together.";
      }
      leaf max-bundle-size {
      type uint32;
      default 400;
      description
      "The maximum number of data objects to be included by the publisher in a si
ngle
       message to recievers.";
     }
     leaf max-bundle-latency {
      type uint32;
      units centiseconds;
      default 400;
      description
      "The maximum latency before a specific YANG Notifications generated
       must egress a publisher.";
     }
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }
    }
  }
}
<CODE ENDS>
```

4. Bulk Notification YANG Module

```
<CODE BEGINS> file "ietf-bulk-notification@2019-09-23.yang"
module ietf-bulk-notification {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bulk-notification";
  prefix bn;

  import ietf-yang-structure-ext {
    prefix sx;
  }
  import ietf-notification-messages {
    prefix nm;
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "";
  description
    "NETCONF Protocol Data Types and Protocol Operations.

     Copyright (c) 2011 IETF Trust and the persons identified as
     the document authors.  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
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC 6241; see
     the RFC itself for full legal notices.";

  revision 2019-09-23 {
    description
      "Initial revision";
    reference
      "RFC XXXX: Bulk Subscription to YANG Event Notification";
  }

  sx:augment-structure "/nm:message/nm:message-header" {
     leaf group-id {
      type string;
      description
        "To identify the group to which a set of notifications belongs.";
    }
```

```
    leaf compression-algorithm {
      type string;
      description
        "The technology with which an originator compress bytestream
         contents.";
    }
    description
      "Group related informations are added to the 'message-header' of
       the ietf-notification-messages to identify the group to which a
       set of notifications belongs and compression algorithms used by
       the publisher.";
  }
}
<CODE ENDS>
```

5. IANA Considerations

5.1. Updates to the IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-bulk-subscription
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-bulk-notification
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

5.2. Updates to the YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC7950]. . Following the format in [RFC6020], the following registration has been made:

```
------------------------------------------------------------------
   Name:         ietf-bulk-subscription
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-bulk-subscription
   Prefix:       trig
   Reference:    RFC xxxx

   Name:         ietf-bulk-notification
   Namespace:    urn:ietf:params:xml:ns:yang:ietf-bulk-notification
   Prefix:       evt
   Reference:    RFC xxxx
------------------------------------------------------------------
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF 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 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:

- /bundle-groups/bundle-group/group-id
- /sn:subscriptions/sn:subscription/bs:max-bundle-latency
- /sn:subscriptions/sn:subscription/bs:max-bundle-size
- /bundle-subscription/input/group-id

7. Acknowledgements

Thanks to Eric Voit, Hui Cai,Zitao Wang for reviewing this draft and providing important input to this document.

8. References

8.1. 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>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

[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>.

[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>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

8.2. 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>.

[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>.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com

Yuexia Fu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: yuexiafu@chinamobile.com