NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

YANG Data Types and Groupings for Cryptography
draft-ietf-netconf-crypto-types-18

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:

* "2020-08-20" --> 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 21 February 2021.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 3
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
2. The "ietf-crypto-types" Module 5
  2.1. Data Model Overview 5
  2.2. Example Usage 17
  2.3. YANG Module 24
3. Security Considerations 43
  3.1. No Support for CRMF 43
  3.2. No Support for Key Generation 43
  3.3. Strength of Keys Configured 44
  3.4. Deletion of Cleartext Key Values 44
  3.5. The "ietf-crypto-types" YANG Module 44
4. IANA Considerations 45
  4.1. The "IETF XML" Registry 45
  4.2. The "YANG Module Names" Registry 46
5. References 46
  5.1. Normative References 46
  5.2. Informative References 47
Appendix A. Change Log 50
  A.1. I-D to 00 50
  A.2. 00 to 01 50
  A.3. 01 to 02 50
  A.4. 02 to 03 50

A.5. 03 to 04 51
A.6. 04 to 05 51
A.7. 05 to 06 51
A.8. 06 to 07 52
A.9. 07 to 08 52
A.10. 08 to 09 52
A.11. 09 to 10 53
A.12. 10 to 11 53
A.13. 11 to 12 53
A.14. 12 to 13 53
A.15. 13 to 14 54
A.16. 14 to 15 54
A.17. 15 to 16 55
A.18. 16 to 17 55
A.19. 17 to 18 55
Acknowledgements 55
Author's Address 56

1. Introduction

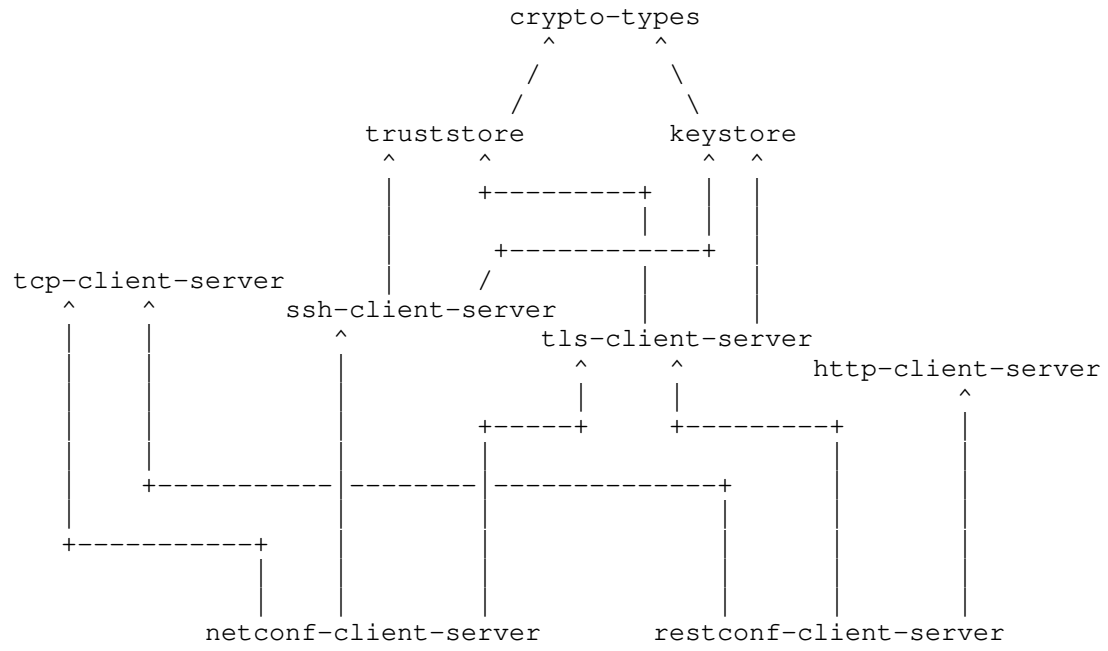
This document presents a YANG 1.1 [RFC7950] module defining identities, typedefs, and groupings useful to cryptographic applications.

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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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 called "ietf-crypto-types". A high-level overview of the module is provided in Section 2.1. Examples illustatrating 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, indentities, typedefs, and groupings.

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
  +-- certificate-expiration-notification
```

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

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}?
```

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

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

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

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
  +-- password-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
```

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

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 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 "password-grouping" Grouping

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

```
  grouping password-grouping
    +-- (password-type)
       +--:(cleartext-password)
       |  +-- cleartext-password?   string
       +--:(encrypted-password) {password-encryption}?
          +-- encrypted-password
             +---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 password-grouping
    +-- (password-type)
       +--:(cleartext-password)
       |  +-- cleartext-password?   string
       +--:(encrypted-password) {password-encryption}?
          +-- encrypted-password
             +-- encrypted-by
             +-- encrypted-value    binary
```

Comments:

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

  - The "cleartext-password" node can encode any plain-text value.
  - The "encrypted-password" node's structure is discussed in Section 2.1.4.1, and is encoded using the CMS EnvelopedData structure.

2.1.4.3. 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) {symmetric-key-encryption}?
          +-- 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) {symmetric-key-encryption}?
          +-- 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 "cleartext-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.

2.1.4.4. 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.

2.1.4.5. 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) {private-key-encryption}?
          +-- 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) {private-key-encryption}?
          +-- encrypted-private-key
             +-- encrypted-by
             +-- encrypted-value    binary
```

   Comments:

   * For the referenced grouping statement(s):

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

2.1.4.6. 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 2.1.4.7 and Section 2.1.4.8.

* 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.

2.1.4.7. 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
            {certificate-expiration-notification}?
       +-- expiration-date    yang:date-and-time
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.6.

* 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 "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
            {certificate-expiration-notification}?
       +-- expiration-date    yang:date-and-time
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.6.

* 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.

2.1.4.9. 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 2.1.4.10 and Section 2.1.4.11.

* 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.

2.1.4.10. 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) {private-key-encryption}?
    |     +-- encrypted-private-key
    |        +-- encrypted-by
    |        +-- encrypted-value    binary
    +-- cert-data?                               end-entity-cert-cms
    +---n certificate-expiration
    |       {certificate-expiration-notification}?
    |  +-- 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 2.1.4.5.
  - The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.8.
  - The "generate-csr-grouping" grouping is discussed in Section 2.1.4.9.

2.1.4.11. 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) {private-key-encryption}?
    |     +-- encrypted-private-key
    |        +-- encrypted-by
    |        +-- encrypted-value    binary
    +-- certificates
    |  +-- certificate* [name]
    |     +-- name?                       string
    |     +-- cert-data                   end-entity-cert-cms
    |     +---n certificate-expiration
    |              {certificate-expiration-notification}?
    |        +-- 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 2.1.4.5.

- The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.8.
- The "generate-csr-grouping" grouping is discussed in Section 2.1.4.9.

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 2.1.4.3) and the "asymmetric-key-pair-with-certs-grouping" (Section 2.1.4.11) 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 "2020-08-20" {
    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) {symmetric-key-encryption}?
  |           +--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) {private-key-encryption}?
            |     +--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
            |             {certificate-expiration-notification}?
            |        +-- 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 2.1.4.9, 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 2.1.4.6, 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@2020-08-20.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 2020-08-20 {
       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.";
      }

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

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

      feature symmetric-key-encryption {
        description
          "Indicates that the server supports password
           encryption.";
      }

      feature private-key-encryption {
        description
          "Indicates that the server supports password
```

```
          encryption.";
      }

      /************************************************/
      /*   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   */
      /*****************/

      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 value, encrypted using the referenced symmetric
             or asymmetric key.";
        }
      }

      grouping password-grouping {
        description
          "A password that MAY be encrypted.";
        choice password-type {
          nacm:default-deny-write;
          mandatory true;
          description
            "Choice between password types.";
          case cleartext-password {
            leaf cleartext-password {
              nacm:default-deny-all;
              type string;
              description
                "The cleartext value of the password.";
```

```
          }
        }
        case encrypted-password {
          if-feature password-encryption;
          container encrypted-password {
            description
             "A container for the encrypted password value.  The
              format of the 'encrypted-value' node is a CMS
              EnvelopedData structure, per Section 8 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).";
            uses encrypted-key-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.";
      }
      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 {
          if-feature symmetric-key-encryption;
          container encrypted-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-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 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 asymmetric-key-pair-grouping {
      description
        "A private key and its associated public key.  Implementations
         SHOULD ensure that the two keys are a matching pair.";
      uses public-key-grouping;
      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.";
      }
      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 {
          if-feature private-key-encryption;
          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-key-value-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.";
        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 values are the "cleartext-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 2.1.4.5).

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 "cleartext-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 "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.

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 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., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[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-04, 8 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-20, 8 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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.

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.

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, Sandra Murphy, Tom Petch, and Wang Haiguang.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

T. Zhou
G. Zheng
Huawei
E. Voit
Cisco Systems
T. Graf
Swisscom
P. Francois
INSA-Lyon

November 02, 2020

Subscription to Distributed Notifications
draft-ietf-netconf-distributed-notif-01

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 May 6, 2021.

Table of Contents

1. Introduction 2
2. Terminologies 3
3. Motivation 4
4. Solution Overview 4
5. Subscription Decomposition 6
6. Publication Composition 6
7. Subscription State Change Notifications 7
8. Publisher Configurations 7
9. YANG Tree 7
10. YANG Module 8
11. IANA Considerations 10
12. Security Considerations 10
13. Contributors 11
14. Acknowledgements 11
15. References 11
15.1. Normative References 11
15.2. Informative References 12
Appendix A. Examples 13
A.1. Dynamic Subscription 13
A.2. Configured Subscription 17
Authors' Addresses 19

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.

Observation Domain: An Observation Domain is the largest set of Observation Points for which metrics can be collected by a metering process. For example, a router line card may be an Observation Domain if it is composed of several interfaces, each of which is an Observation Point. In the YANG notification messages it generates, the Observation Domain includes its Observation Domain ID, which is unique per publisher process. That way, the collecting process can identify the specific Observation Domain from the publisher that sends the YANG notification messages. Every Observation Point is associated with an Observation Domain.

Observation Domain ID: A 32-bit identifier of the Observation Domain that is locally unique to the publisher process. The publisher process uses the Observation Domain ID to uniquely identify to the collecting process the Observation Domain that meters the metrics. Receivers SHOULD use the transport session and the Observation Domain ID field to separate different publisher streams originating from the same publisher.

3. Motivation

Lost and corrupt YANG notification messages need to be recognized at the receiver to ensure data integrity even when multiple publisher processes publishing from the same transport session.

To preserve data integrity down to the publisher process, the Observation Domain ID in the transport message header of the YANG notification message is introduced. In case of UDP transport, this is described in Section 3.2 of UDP based transport [I-D.ietf-netconf-udp-notif].

4. 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.

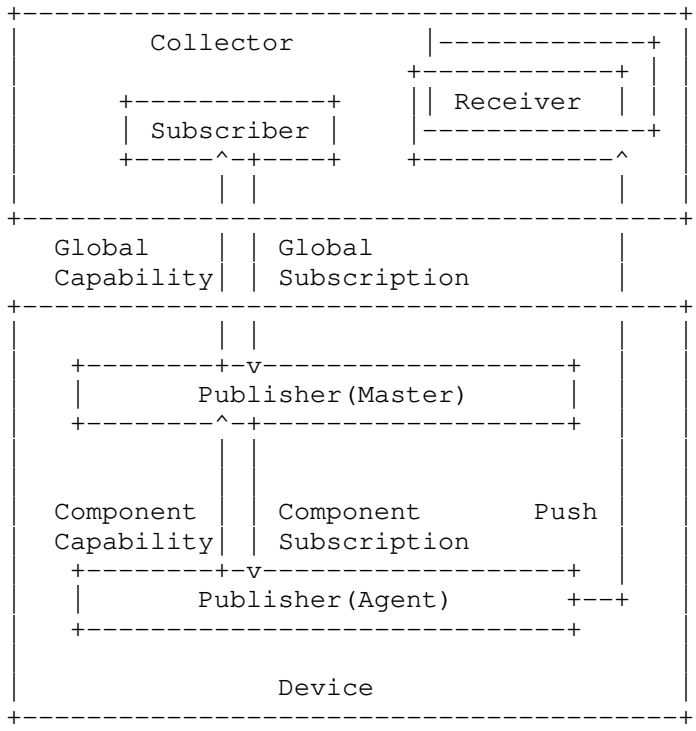


Fig. 2 The Distributed Data Export Framework

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.

5. 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.

6. 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 Observation Domain 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 Observation Domain 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 Observation Domain IDs to notify the current Publishers for the corresponding Global Subscription.

7. 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.

8. 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.ietf-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.

9. YANG Tree

```
module: ietf-distributed-notifications
  augment /sn:subscriptions/sn:subscription:
    +--ro message-observation-domain-id*   string
  augment /sn:subscription-started:
    +--ro message-observation-domain-id*   string
  augment /sn:subscription-modified:
    +--ro message-observation-domain-id*   string
  augment /sn:establish-subscription/sn:output:
    +--ro message-observation-domain-id*   string
```

10. 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-observation-domain-ids {
      description
        "Provides a reusable list of message-observation-domain-ids.";

      leaf-list message-observation-domain-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
        Observation Domain ID to be exposed for a subscription.";

      uses message-observation-domain-ids;
    }

    augment "/sn:subscription-started" {
      description
        "This augmentation allows MSO specific parameters to be
         exposed for a subscription.";

      uses message-observation-domain-ids;
    }

    augment "/sn:subscription-modified" {
      description
        "This augmentation allows MSO specific parameters to be
         exposed for a subscription.";

      uses message-observation-domain-ids;
    }
```

```
     augment "/sn:establish-subscription/sn:output" {
       description
         "This augmentation allows MSO specific parameters to be
          exposed for a subscription.";

       uses message-observation-domain-ids;
     }
   }
   <CODE ENDS>
```

11. 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

12. 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:

- /subscriptions/subscription/message-observation-domain-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].

13. Contributors

Alexander Clemm
Futurewai
2330 Central Expressway
Santa Clara
California
United States of America
Email: ludwig@clemm.org

14. Acknowledgements

We thank Kent Watsen, Mahesh Jethanandani, Martin Bjorklund, Tim Carey and Qin Wu for their constructive suggestions for improving this document.

15. References

15.1. 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>.

15.2. 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-05 (work in progress), October 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.ietf-netconf-udp-notif]
Zhou, T., Zheng, G., Lucente, P., Graf, T., and P. Francois, "UDP-based Transport for Configured Subscriptions", draft-ietf-netconf-udp-notif-01 (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.

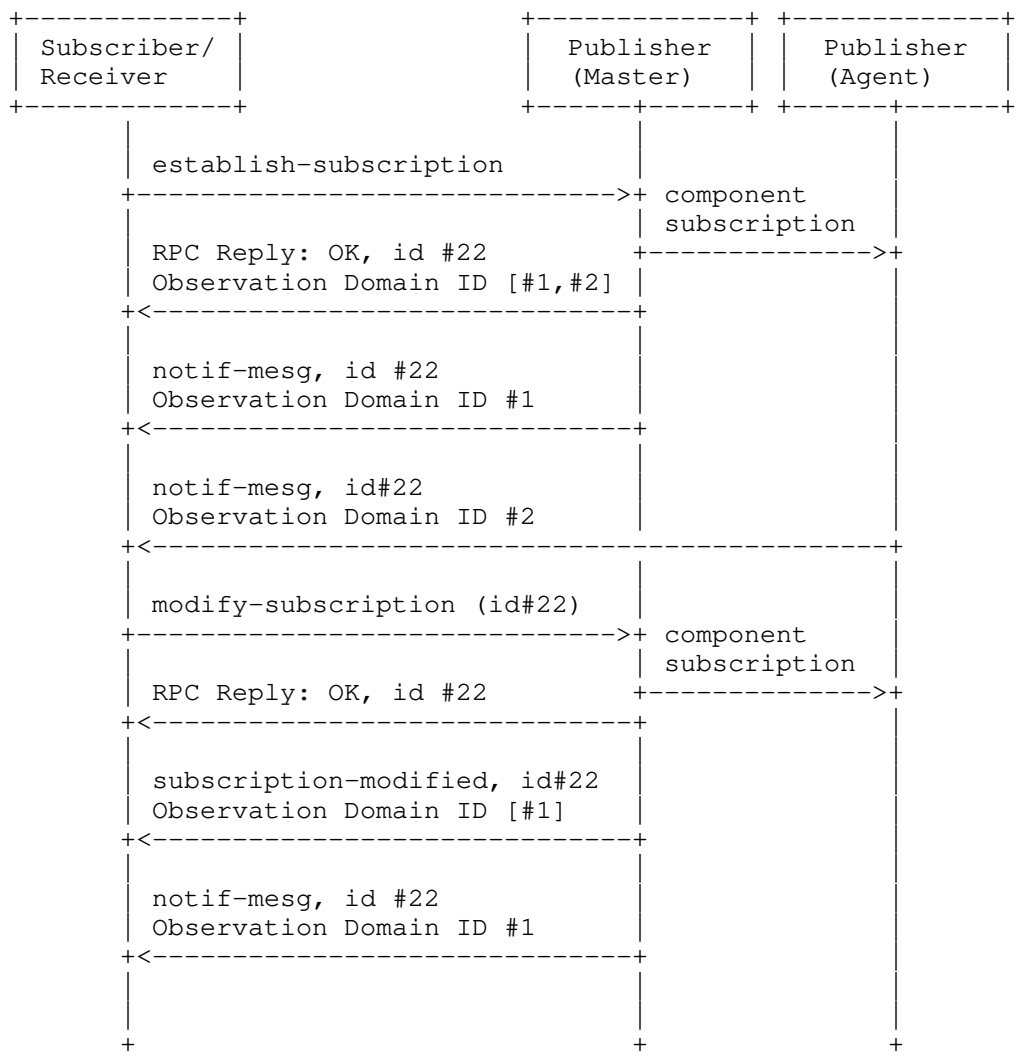


Fig. 3 Call Flow for Dynamic Subscription

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 Observation Domain ID: #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-observation-domain-id
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
       1
    </message-observation-domain-id>
    <message-observation-domain-id
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
       2
    </message-observation-domain-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 Observation Domain #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-observation-domain-id
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notificationss>
        1
     </message-observation-domain-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.

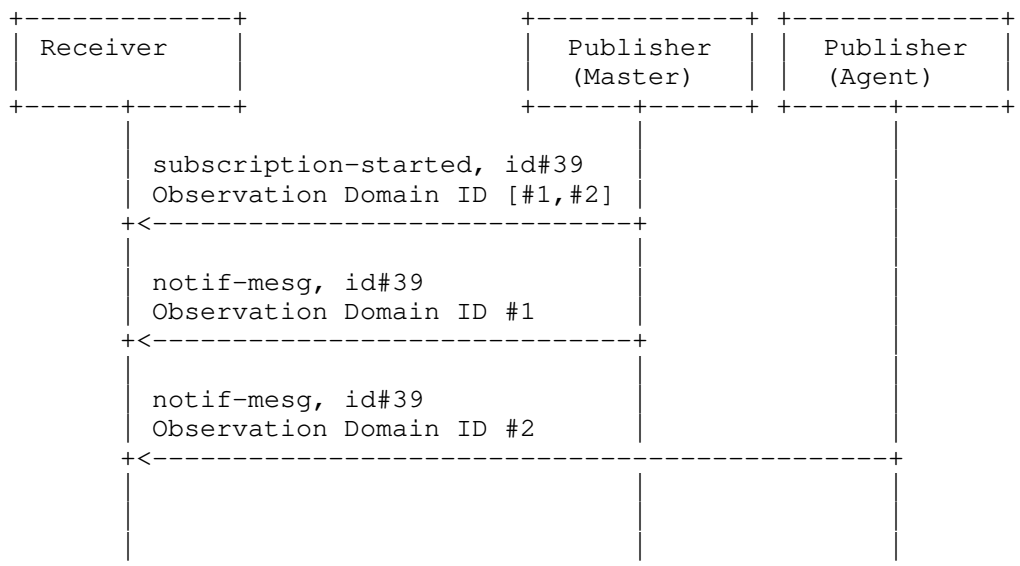


Fig. 8 Call Flow for Configured Subscription

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 Observation Domain: #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-observation-domain-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
       1
    </message-observation-domain-id>
    <message-observation-domain-id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications>
       2
    </message-observation-domain-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 Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

YANG Groupings for HTTP Clients and HTTP Servers
draft-ietf-netconf-http-client-server-05

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:

* "2020-08-20" --> 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 21 February 2021.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 3
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
2. The "ietf-http-client" Module 5
  2.1. Data Model Overview 6

2.2. Example Usage 9
2.3. YANG Module 10
3. The "ietf-http-server" Module 17
3.1. Data Model Overview 17
3.2. Example Usage 19
3.3. YANG Module 19
4. Security Considerations 24
4.1. The "ietf-http-client" YANG Module 24
4.2. The "ietf-http-server" YANG Module 25
5. IANA Considerations 26
5.1. The "IETF XML" Registry 26
5.2. The "YANG Module Names" Registry 26
6. References 27
6.1. Normative References 27
6.2. Informative References 27
Appendix A. Change Log 29
A.1. 00 to 01 29
A.2. 01 to 02 30
A.3. 02 to 03 30
A.4. 03 to 04 30
A.5. 04 to 05 30
Acknowledgements 31
Author's Address 31

1. Introduction

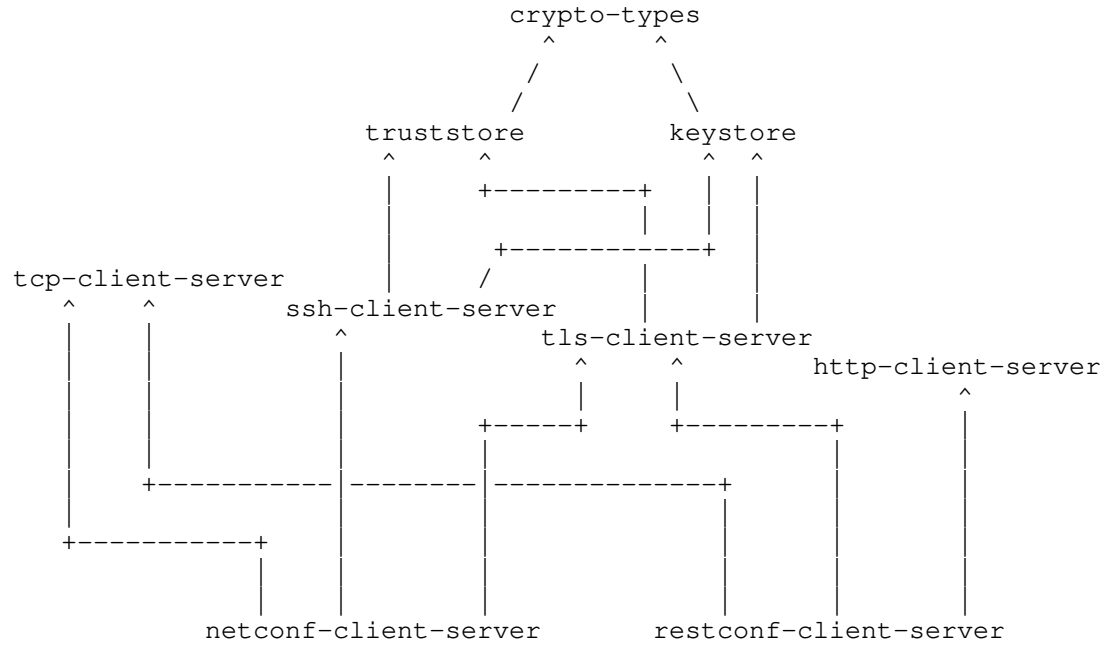
This document defines two YANG 1.1 [RFC7950] modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based 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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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

This section defines a YANG 1.1 [RFC7950] module called "ietf-http-client". A high-level overview of the module is provided in Section 2.1. Examples illustatrating 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:
  +-- proxy-connect
  +-- basic-auth
  +-- tcp-supported
  +-- tls-supported
```

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

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
```

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

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 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>
      <cleartext-password>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 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>
      <cleartext-password>secret</cleartext-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>
              <cleartext-password>secret</cleartext-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@2020-08-20.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";
     }

     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 2020-08-20 {
      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.";
            }
            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 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 {
                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;
                }
              }
```

```
          }
          case https {
            container https-proxy {
              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

This section defines a YANG 1.1 [RFC7950] module called "ietf-http-server". A high-level overview of the module is provided in Section 3.1. Examples illustatrating 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-config-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 following diagram lists all the "grouping" statements defined in the "ietf-http-server" module:

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

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

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@2020-08-20.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 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";
    }

    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 2020-08-20 {
      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 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

[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., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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.

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.

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
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2021

M. Jethanandani
Kloud Services
K. Watsen
Watsen Networks

October 23, 2020

An HTTPS-based Transport for Configured Subscriptions
draft-ietf-netconf-https-notif-05

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 April 26, 2021.

Table of Contents

1. Introduction 2
1.1. Applicability Statement 3
1.2. Note to RFC Editor 3
1.3. Abbreviations 3
1.4. Terminology 4
1.4.1. Subscribed Notifications 4
1.5. Receiver and Publisher Interaction 4
1.5.1. Pipelining of messages 4
2. Learning Receiver Capabilities 7
2.1. Introduction 7
2.2. Example 7
3. The "ietf-sub-notif-recv-list" Module 8
3.1. Data Model Overview 8
3.2. YANG Module 8
4. The "ietf-https-notif" Module 10
4.1. Data Model Overview 10
4.2. YANG module 11
5. Security Considerations 14
6. Encoding Event Notifications 14
7. IANA Considerations 16
7.1. URI Registration 16
7.2. YANG Module Name Registration 16
7.3. Media Types 16
7.3.1. Media Type "application/ietf-https-notif-cap+xml . . 16
7.3.2. Media Type "application/ietf-https-notif-cap+json . . 17
8. Examples 19
8.1. Subscribed Notification based Configuration 19
8.2. Non Subscribed Notification based Configuration 21
8.3. Bundled Message 24
9. Contributors 25
10. Acknowledgements 25
11. Normative references 25
Authors' Addresses 27

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.

2020-10-21 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].

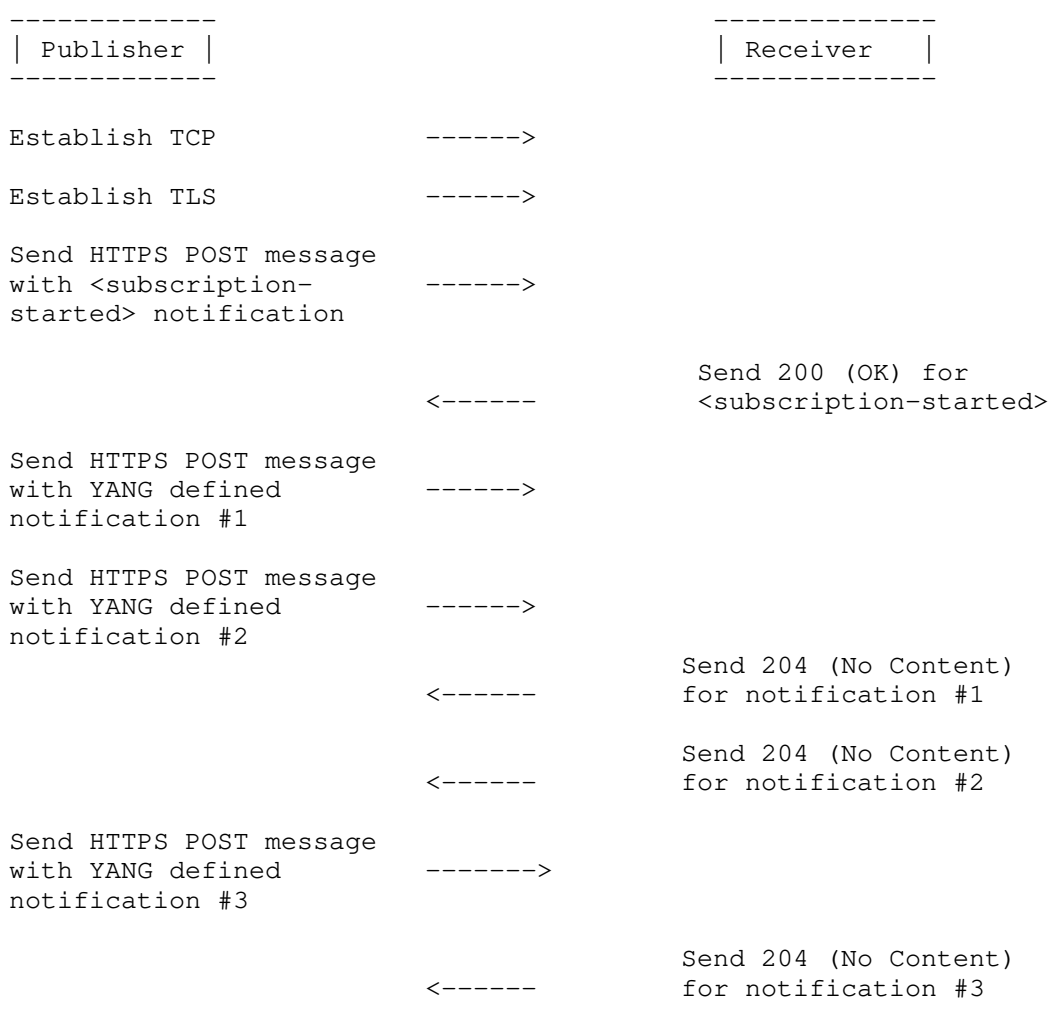
- Subscribed Notifications

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. This example shows the flow assuming that Subscribed Notifications is used and therefore a <subscription-started> notification is sent before sending the first notification. The example would be the same for when Subscribed Notification is not used by removing the first POST message for <susbscription-started>.



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@2020-10-21.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 "2020-10-21" {
    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@2020-10-21.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 "2020-10-21" {
    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. Encoding Event Notifications

Notifications are encoded as defined in RESTCONF [RFC8040] Section 6.4. The examples in that section apply for sending notifications over the "https-notif" based transport.

An example of YANG-Push in JSON would look something like this:

```
data: {
data:   "ietf-restconf:notification" : {
data:     "eventTime" : "2017-10-25T08:00:11.22Z",
data:     "ietf-yang-push:push-update" : {
data:       "id" : 1011,
data:       "datastore-contents" : {
data:         "ietf-interfaces:interfaces" : {
data:           "interface": [
data:             {
data:               "name": "eth0",
data:               "oper-status": "up"
data:             }
data:           ],
data:         },
data:         "eventClass" : "state",
data:         "reportingEntity": {
data:           "card": "Ethernet0"
data:         }
data:         "severity" : "minor"
data:       }
data:     }
data:   }
data: }
```

An example of YANG-Push in XML would look something like this:

```
data: <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
data:  <eventTime>2017-10-25T08:00:11.22Z</eventTime>
data:  <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
data:   <id>1011</id>
data:   <datastore-contents>
data:     <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
data:      <interface>
data:       <name>eth0</name>
data:        <oper-status>up</oper-status>
data:       </interface>
data:      </interfaces>
data:      <eventClass>state</eventClass>
data:      <reportingEntity>
data:       <card>Ethernet0</card>
data:      </reportingEntity>
data:      <serverity>minor</serverity>
data:    </datastore-contents>
data:   </push-update>
data:  </notification>
```

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>
                  <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>
                  <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>
      <transport xmlns:hn="urn:ietf:params:xml:ns:yang:ietf-https-no\
tif">
        hn:https
      </transport>
      <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">
    <certificate-bags>
      <certificate-bag>
        <name>explicitly-trusted-server-ca-certs</name>
        <description>
          Trust anchors (i.e. CA certs) that are used to authenticat\
e
          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-data>base64encodedvalue==</cert-data>
        </certificate>
        <certificate>
          <name>Fred Flintstone</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificate-bag>
    </certificate-bags>
  </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 "2020-10-21" {
    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'.";
        }
      }
    }
  }
```

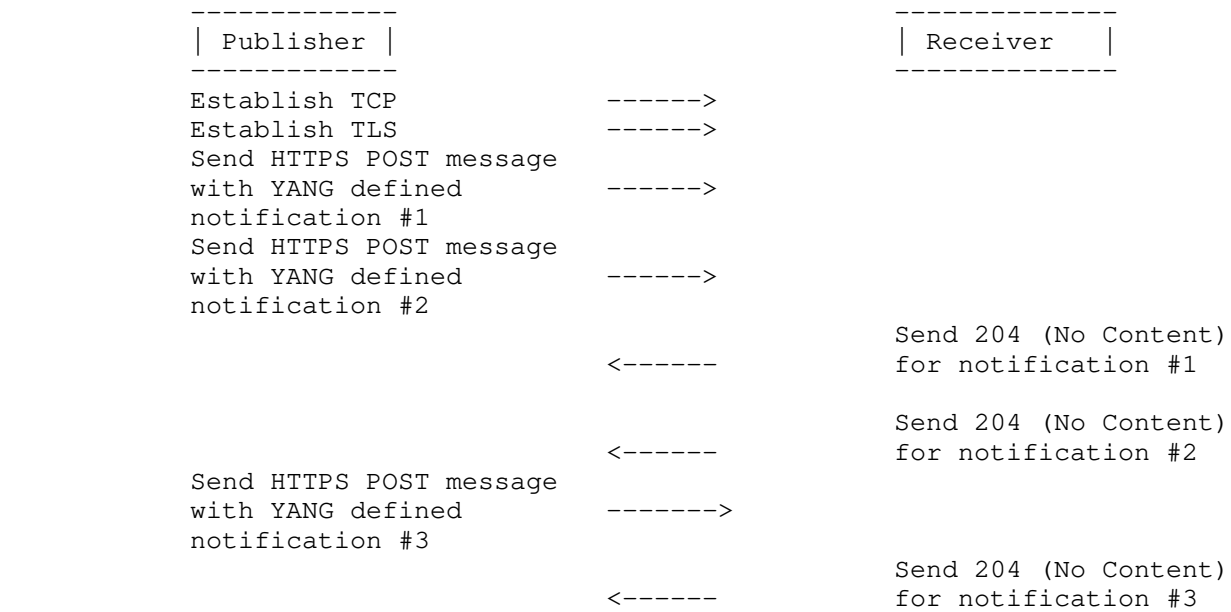
This example shows how a HTTPS client can be configured to send notifications to a receiver at address 192.0.2.1, port 443, a 'path', with server certificates, and the corresponding trust store that is used to authenticate a connection.

```
  [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>
                  <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>
                  <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">
      <certificate-bags>
        <certificate-bag>
          <name>explicitly-trusted-server-ca-certs</name>
          <description>
            Trust anchors (i.e. CA certs) that are used to authenticat\
  e
            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-data>base64encodedvalue==</cert-data>
          </certificate>
          <certificate>
            <name>Fred Flintstone</name>
            <cert-data>base64encodedvalue==</cert-data>
          </certificate>
        </certificate-bag>
      </certificate-bags>
    </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-05 (work in progress), August 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 Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

A YANG Data Model for a Keystore
draft-ietf-netconf-keystore-20

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 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:

* "2020-08-20" --> 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.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 4
  1.2. Specification Language 5
  1.3. Terminology 5
  1.4. Adherence to the NMDA 6
2. The "ietf-keystore" Module 6
  2.1. Data Model Overview 6
  2.2. Example Usage 17
  2.3. YANG Module 29
3. Support for Built-in Keys 37
4. Encrypting Keys in Configuration 40
5. Security Considerations 44
  5.1. Data at Rest 44
  5.2. The "ietf-keystore" YANG Module 44
6. IANA Considerations 45
  6.1. The "IETF XML" Registry 45
  6.2. The "YANG Module Names" Registry 45
7. References 45
  7.1. Normative References 45
  7.2. Informative References 46

Appendix A. Change Log 48
A.1. 00 to 01 48
A.2. 01 to 02 48
A.3. 02 to 03 48
A.4. 03 to 04 48
A.5. 04 to 05 49
A.6. 05 to 06 49
A.7. 06 to 07 49
A.8. 07 to 08 49
A.9. 08 to 09 49
A.10. 09 to 10 50
A.11. 10 to 11 50
A.12. 11 to 12 50
A.13. 12 to 13 51
A.14. 13 to 14 51
A.15. 14 to 15 51
A.16. 15 to 16 51
A.17. 16 to 17 51
A.18. 17 to 18 52
A.19. 18 to 19 52
A.20. 19 to 20 52
Acknowledgements 52
Author's Address 52

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 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 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 Platform Module (TPM). These systems are unique in that the cryptographic hardware hides the secret key values. Additionally, such hardware is commonly initiailized when manufactured to protect a "built-in" asymmetric key for which the public half is conveyed in an identity certificate (e.g., an IDevID [Std-802.1AR-2009] certificate). Please see Section 3 to see how built-in keys are supported.

This document intends to support existing practices; it does not intend to define new behvior for systems to implement. To simplify implementation, advanced key formats may be selectively implemented.

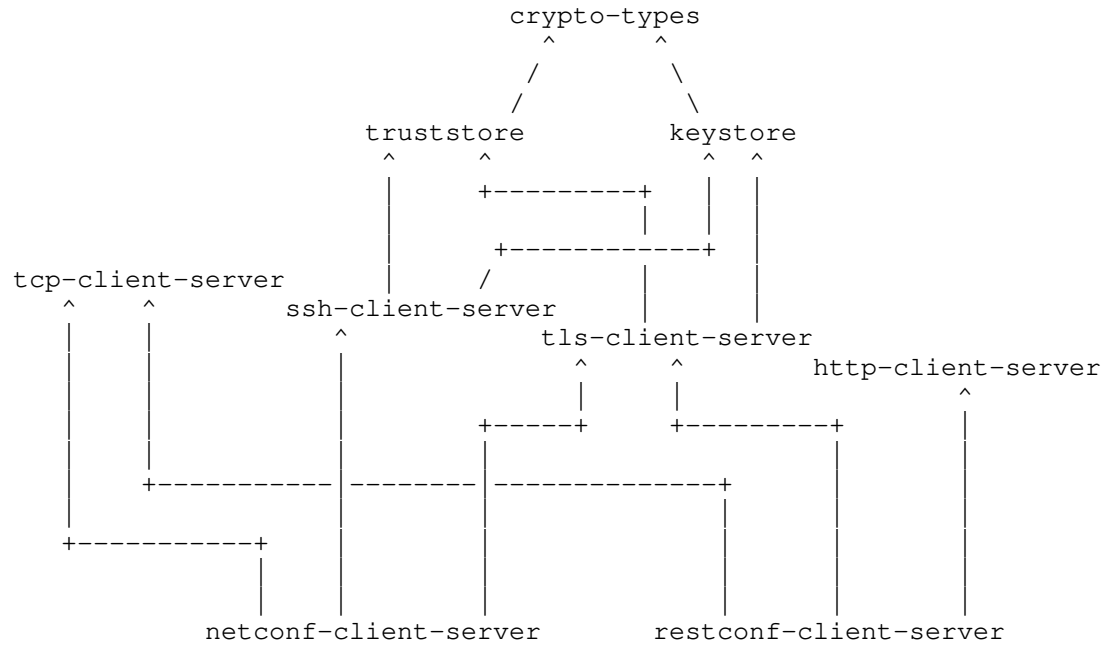
Implementations may utilize zero or more operating system level keystore utilities and/or hardware security modules (HSMs).

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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Terminology

The terms "client" and "server" are defined in [RFC6241] and are not redefined here.

The term "keystore" is defined in this draft as a mechanism that intends safeguard secrets placed into it for protection.

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].

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).

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 illustatrating 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-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:
  +-- keystore-supported
  +-- local-definitions-supported
```

```
  | 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
    +-- symmetric-key-ref
    +-- asymmetric-key-ref
```

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

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
```

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

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?    ks:symmetric-key-ref
     +--:(asymmetric-key-ref)
        +-- asymmetric-key-ref?   ks: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.

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 for whether a 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 2.1.4.3 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 for whether an asymmetric key is defined locally 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 "local-definition" option, the defintion uses the "asymmetric-key-pair-grouping" grouping discussed in Section 2.1.4.5 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 for whether an asymmetric key is defined locally 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 "local-definition" option, the defintion uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 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 for whether a 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 2.1.4.11 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 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 in Section 2.1.4.11 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
     +---u keystore-grouping

  grouping encrypted-by-choice-grouping
    +-- (encrypted-by-choice)
       +--:(symmetric-key-ref)
       |  +-- symmetric-key-ref?    ks:symmetric-key-ref
```

```
        +--:(asymmetric-key-ref)
           +-- asymmetric-key-ref?   ks:asymmetric-key-ref
  grouping asymmetric-key-certificate-ref-grouping
    +-- asymmetric-key?   ks:asymmetric-key-ref
    +-- certificate?      leafref
  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
  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
  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
  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
  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
```

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:

```
   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) {private-key-encryption}?
        |     |     +--rw encrypted-private-key
        |     |        +--rw encrypted-by
        |     |        |  +--rw (encrypted-by-choice)
        |     |        |     +--:(symmetric-key-ref)
        |     |        |     |  +--rw symmetric-key-ref?
        |     |        |     |           ks:symmetric-key-ref
        |     |        |     +--:(asymmetric-key-ref)
        |     |        |        +--rw asymmetric-key-ref?
        |     |        |                 ks:asymmetric-key-ref
        |     |        +--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-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) {symmetric-key-encryption}?
                    +--rw encrypted-key
```

```
                      +--rw encrypted-by
                      |  +--rw (encrypted-by-choice)
                      |     +--:(symmetric-key-ref)
                      |     |  +--rw symmetric-key-ref?
                      |     |           ks:symmetric-key-ref
                      |     +--:(asymmetric-key-ref)
                      |        +--rw asymmetric-key-ref?
                      |                 ks:asymmetric-key-ref
                      +--rw encrypted-value    binary

    grouping encrypted-by-choice-grouping
      +-- (encrypted-by-choice)
         +--:(symmetric-key-ref)
         |  +-- symmetric-key-ref?    ks:symmetric-key-ref
         +--:(asymmetric-key-ref)
            +-- asymmetric-key-ref?   ks:asymmetric-key-ref
    grouping asymmetric-key-certificate-ref-grouping
      +-- asymmetric-key?   ks:asymmetric-key-ref
      +-- certificate?      leafref
    grouping local-or-keystore-symmetric-key-grouping
      +-- (local-or-keystore)
         +--:(local) {local-definitions-supported}?
         |  +-- local-definition
         |     +-- key-format?            identityref
         |     +-- (key-type)
         |        +--:(cleartext-key)
         |        |  +-- cleartext-key?   binary
         |        +--:(hidden-key)
         |        |  +-- hidden-key?      empty
         |        +--:(encrypted-key) {symmetric-key-encryption}?
         |           +-- encrypted-key
         |              +-- encrypted-by
         |              +-- encrypted-value    binary
         +--:(keystore) {keystore-supported}?
            +-- keystore-reference?   ks:symmetric-key-ref
    grouping local-or-keystore-asymmetric-key-grouping
      +-- (local-or-keystore)
         +--:(local) {local-definitions-supported}?
         |  +-- local-definition
         |     +-- 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) {private-key-encryption}?
```

```
          |           +-- encrypted-private-key
          |              +-- encrypted-by
          |              +-- encrypted-value    binary
          +--:(keystore) {keystore-supported}?
             +-- keystore-reference?   ks:asymmetric-key-ref
       grouping local-or-keystore-asymmetric-key-with-certs-grouping
         +-- (local-or-keystore)
            +--:(local) {local-definitions-supported}?
            |  +-- local-definition
            |     +-- 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) {private-key-encryption}?
            |     |     +-- encrypted-private-key
            |     |        +-- encrypted-by
            |     |        +-- encrypted-value    binary
            |     +-- certificates
            |     |  +-- certificate* [name]
            |     |     +-- name?                     string
            |     |     +-- cert-data                 end-entity-cert-cms
            |     |     +---n certificate-expiration
            |     |             {certificate-expiration-notification}?
            |     |        +-- 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}?
               +-- keystore-reference?   ks:asymmetric-key-ref
       grouping local-or-keystore-end-entity-cert-with-key-grouping
         +-- (local-or-keystore)
            +--:(local) {local-definitions-supported}?
            |  +-- local-definition
            |     +-- 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) {private-key-encryption}?
          |      |     +-- encrypted-private-key
          |      |        +-- encrypted-by
          |      |        +-- encrypted-value    binary
          |      +-- cert-data?
          |      |       end-entity-cert-cms
          |      +---n certificate-expiration
          |      |       {certificate-expiration-notification}?
          |      |  +-- 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}?
             +-- keystore-reference
                +-- asymmetric-key?   ks:asymmetric-key-ref
                +-- certificate?      leafref
    grouping keystore-grouping
      +-- asymmetric-keys
      |  +-- asymmetric-key* [name]
      |     +-- name?                                  string
      |     +-- 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) {private-key-encryption}?
      |     |     +-- encrypted-private-key
      |     |        +-- encrypted-by
      |     |        +-- encrypted-value    binary
      |     +-- certificates
      |     |  +-- certificate* [name]
      |     |     +-- name?                     string
      |     |     +-- cert-data                 end-entity-cert-cms
      |     |     +---n certificate-expiration
      |     |             {certificate-expiration-notification}?
      |     |        +-- 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
      |
```

```
      +-- symmetric-keys
         +-- symmetric-key* [name]
            +-- name?                  string
            +-- key-format?            identityref
            +-- (key-type)
               +--:(cleartext-key)
               |  +-- cleartext-key?   binary
               +--:(hidden-key)
               |  +-- hidden-key?      empty
               +--:(encrypted-key) {symmetric-key-encryption}?
                  +-- encrypted-key
                     +-- encrypted-by
                     +-- 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 protcol-accessible nodes for the "ietf-keystore" module are an instance of the "keystore-grouping" grouping 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-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.

These examples assume the existence of an example module called "ex-keystore-usage" having the namespace "http://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 "local-or-keystore" groupings in use, followed by the YANG module itself.

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

```
module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                                                string
     |  +---u ks:local-or-keystore-symmetric-key-grouping
     +--rw asymmetric-key* [name]
     |  +--rw name                                                 string
     |  +---u ks:local-or-keystore-asymmetric-key-grouping
     +--rw asymmetric-key-with-certs* [name]
     |  +--rw name
     |  |       string
     |  +---u ks:local-or-keystore-asymmetric-key-with-certs-grouping
     +--rw end-entity-cert-with-key* [name]
        +--rw name
        |       string
        +---u ks:local-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 keystore's full structure to be seen:

```
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) {symmetric-key-encryption}?
     |     |           +--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)
          |      |                 {private-key-encryption}?
          |      |           +--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)
          |     |     |           {private-key-encryption}?
          |     |     |     +--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
          |     |     |             {certificate-expiration-notification}?
          |     |     |        +-- 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)
      |     |           {private-key-encryption}?
      |     |     +--rw encrypted-private-key
      |     |        +--rw encrypted-by
      |     |        +--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-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>
```

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

```
  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 "2020-08-20" {
      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 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@2020-08-20.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 2020-08-20 {
      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 that can be augmented
         into the 'encrypted-by' node presented by the
         'symmetric-key-grouping' and 'asymmetric-key-pair-grouping'
         groupings defined in RFC AAAA.";
      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 ks:symmetric-key-ref;
            description
             "Identifies the symmetric key used to encrypt the
              associated key.";
          }
        }
        case asymmetric-key-ref {
          leaf asymmetric-key-ref {
            type ks:asymmetric-key-ref;
            description
             "Identifies the asymmetric key whose public key
              encrypted the associated 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, i.e., within the using data model,
         or 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, i.e., within the using data model,
         or a reference to an asymmetric 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: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.";
          }
        }
      }
    }

    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,
         i.e., within the using data model, or a reference to an
         asymmetric key (and its associated certificates) 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: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.";
          }
        }
      }
    }

    grouping local-or-keystore-end-entity-cert-with-key-grouping {
      description
        "A grouping that expands to allow an end-entity certificate
         (and its associated asymmetric key pair) to be either stored
         locally, i.e., within the using data model, or a reference
         to a specific certificate 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: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 associated with
               an asymmetric key stored 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 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 primary 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 value for the list's "key" substatement, so that the server can bind the references to the built-in entries.

In addition to copying keys into the Keystore in <running>, cleartext and encrypted keys may be copied into other parts of configuration, but they will lose their connection to having been a built-in value. Note that hidden keys cannot be copied into other parts of the configuration because doing would lose the key's connection to the built-in key, where the key's secret value is stored. Built-in "encrypted" keys MAY be copied into other parts of the configuration so long as the reference to the other built-in key that encrypted them is maintained.

Only the referenced keys need to be copied; that is, the keys in <running> MAY be a subset, including the whole of the set, of the built-in keys defined in <operational>.

No new built-in keys may be added nor existing built-in changed, with exception for associating additional certificates to an existing built-in key.

A server MUST reject attempts to modify any aspect of built-in keys, with exception for 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 both the symmetric and asymmetric keys on a server to be encrypted, such that traditional backup/restore procedures can be used without concern for the keys being compromised when in transit.

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 system. 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 MAY provide an API enabling the encryption of other keys or, alternatively, let the administrators do so themselves using the asymmetric key's public half.

A server MUST possess (or be able to possess, in case the KEK has been encrypted by another KEK) a KEK's cleartext value so that it can decrypt the other keys in the configurion 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 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

Implementations SHOULD provide an API that simultaneously generates and encrypts a key (symmetric or asymmetric) using a KEK. Thusly newly generated key cleartext values are never known to the administrators generating the 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 (e.g., an organization's crypto officer).

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 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 many RMA 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 (e.g., NACM's "default-deny-all), 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 doesn't 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 key, 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 public key 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.

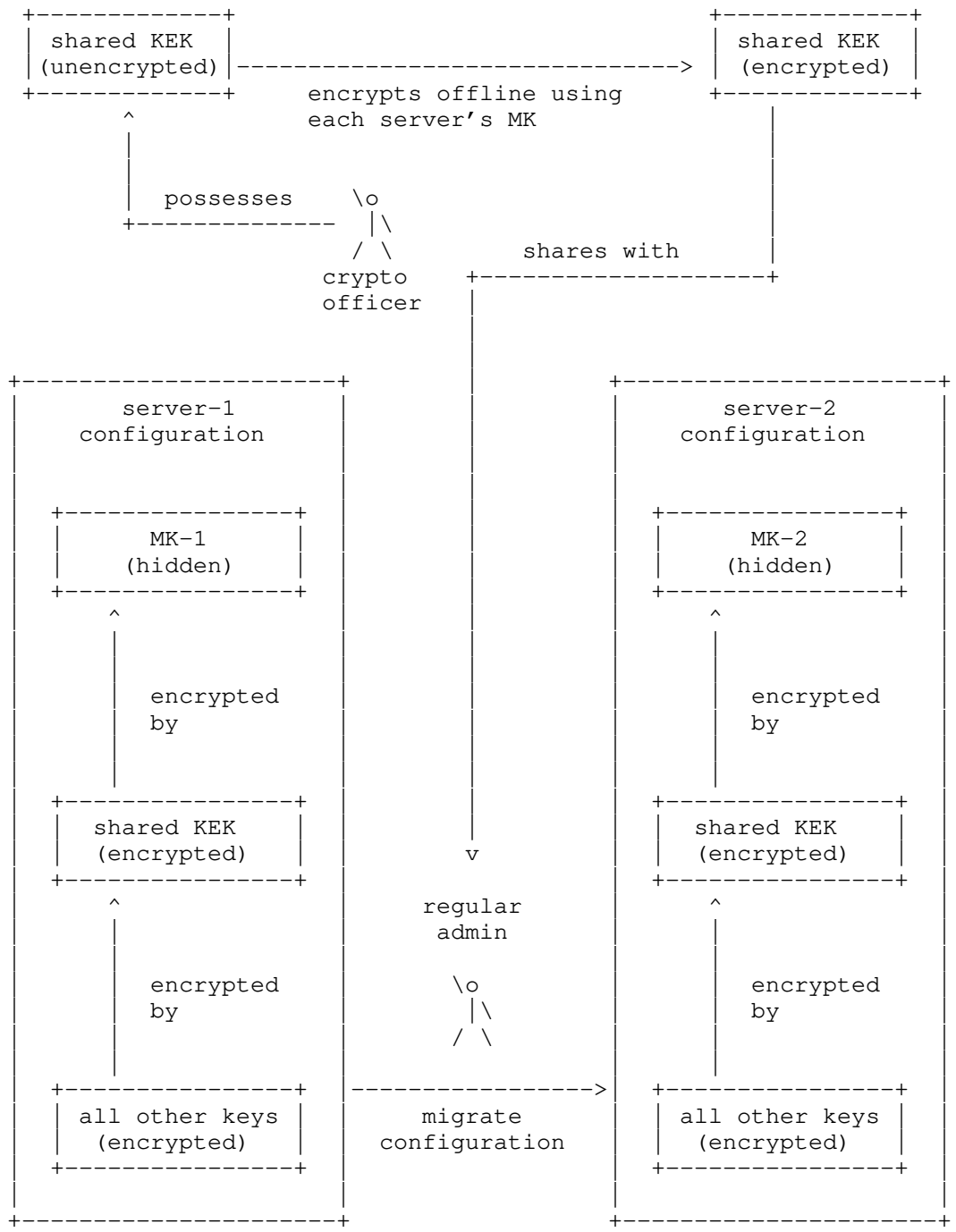
It is highly RECOMMENDED that MKs are built-in and hidden but, if this is not possible, MKs highly restricted access mechanisms SHOULD be used to limit access to the MK's secret data to only highly authorized clients (e.g., an organization's crypto officer). In this case, it is RECOMMENDED that the MK is not built-in and hence is, effectively, just like a KEK.

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 the 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. 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 "cleartext-key" and "cleartext-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 uncontrolled 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 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-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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.

A.20. 19 to 20

* Addressed SecDir comments from Magnus Nystroem and Sandra Murphy.

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, Magnus Nystroem, Martin Bjorklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Reshad Rahman, Sandra Murphy, 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: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-21

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:

* "2020-08-20" --> 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 21 February 2021.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 4
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
2. The "ietf-netconf-client" Module 5
  2.1. Data Model Overview 6

2.2. Example Usage 10
2.3. YANG Module 14
3. The "ietf-netconf-server" Module 25
3.1. Data Model Overview 25
3.2. Example Usage 30
3.3. YANG Module 36
4. Security Considerations 49
4.1. The "ietf-netconf-client" YANG Module 49
4.2. The "ietf-netconf-server" YANG Module 49
5. IANA Considerations 50
5.1. The "IETF XML" Registry 50
5.2. The "YANG Module Names" Registry 50
6. References 51
6.1. Normative References 51
6.2. Informative References 52
Appendix A. Change Log 54
A.1. 00 to 01 54
A.2. 01 to 02 54
A.3. 02 to 03 54
A.4. 03 to 04 54
A.5. 04 to 05 54
A.6. 05 to 06 55
A.7. 06 to 07 55
A.8. 07 to 08 55
A.9. 08 to 09 55
A.10. 09 to 10 55
A.11. 10 to 11 56
A.12. 11 to 12 56
A.13. 12 to 13 56
A.14. 13 to 14 56
A.15. 14 to 15 57
A.16. 15 to 16 57
A.17. 16 to 17 57
A.18. 17 to 18 57
A.19. 18 to 19 57
A.20. 19 to 20 57
A.21. 20 to 21 58
Acknowledgements 58
Author's Address 58

1. Introduction

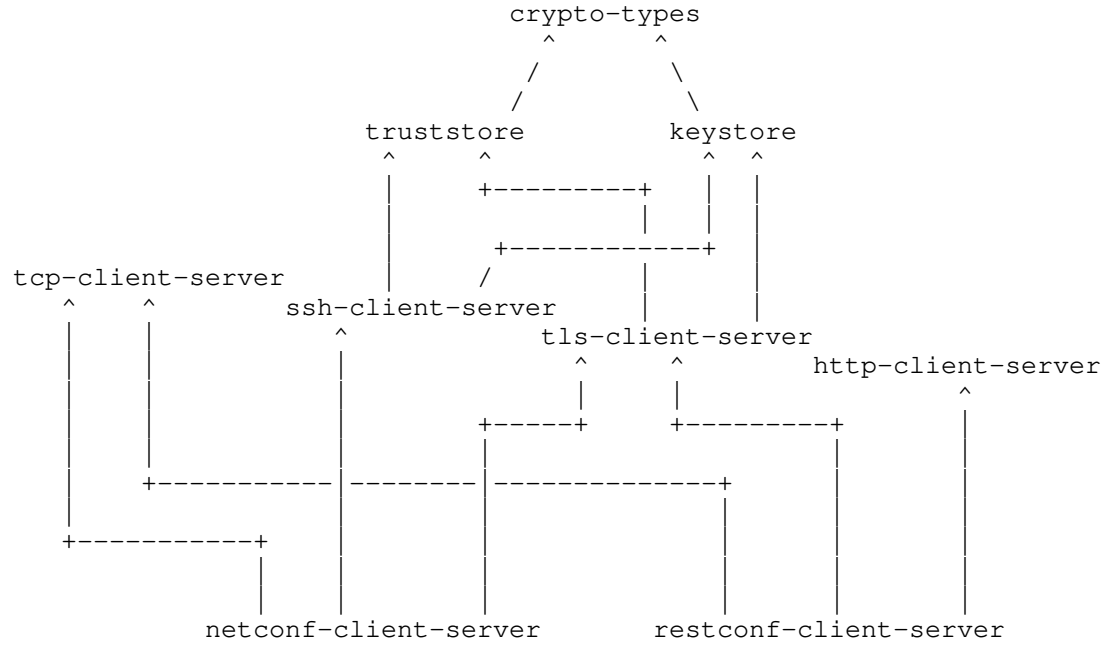
This document defines two YANG [RFC7950] modules, one module to configure a NETCONF [RFC6241] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [RFC6242] and NETCONF over TLS [RFC7589] and NETCONF Call Home connections [RFC8071].

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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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

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
```

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

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
```

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

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 tree diagram [RFC8340] 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\
re-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@2020-08-20.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 2020-08-20; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    revision-date 2020-08-20; // 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 2020-08-20 {
      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

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
```

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

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
```

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

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 tree diagram [RFC8340] 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
```

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

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</truststo\
    ore-reference>
                </ca-certs>
                <ee-certs>
                  <truststore-reference>trusted-client-ee-certs</truststo\
    ore-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@2020-08-20.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 2020-08-20; // stable grouping definitions
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }

     import ietf-ssh-server {
       prefix sshs;
       revision-date 2020-08-20; // stable grouping definitions
       reference
         "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
     }

     import ietf-tls-server {
       prefix tlss;
       revision-date 2020-08-20; // 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 2020-08-20 {
    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 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-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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.

A.21. 20 to 21

* Cleaned up titles in the IANA Considerations section

* Fixed issues found by the SecDir review of the "keystore" 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, 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: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-21

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:

* "2020-08-20" --> 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

This Internet-Draft will expire on 21 February 2021.

Copyright Notice

Table of Contents

1. Introduction 4

1.1. Relation to other RFCs 4
1.2. Specification Language 5
1.3. Adherence to the NMDA 5
2. The "ietf-restconf-client" Module 5
2.1. Data Model Overview 6
2.2. Example Usage 10
2.3. YANG Module 14
3. The "ietf-restconf-server" Module 24
3.1. Data Model Overview 24
3.2. Example Usage 29
3.3. YANG Module 33
4. Security Considerations 45
4.1. The "ietf-restconf-client" YANG Module 45
4.2. The "ietf-restconf-server" YANG Module 46
5. IANA Considerations 46
5.1. The "IETF XML" Registry 46
5.2. The "YANG Module Names" Registry 47
6. References 47
6.1. Normative References 47
6.2. Informative References 48
Appendix A. Expanded Tree Diagrams 50
A.1. Expanded Tree Diagram for 'ietf-restconf-client' 50
A.2. Expanded Tree Diagram for 'ietf-restconf-server' 50
Appendix B. Change Log 50
B.1. 00 to 01 50
B.2. 01 to 02 51
B.3. 02 to 03 51
B.4. 03 to 04 51
B.5. 04 to 05 51
B.6. 05 to 06 51
B.7. 06 to 07 52
B.8. 07 to 08 52
B.9. 08 to 09 52
B.10. 09 to 10 52
B.11. 10 to 11 52
B.12. 11 to 12 53
B.13. 12 to 13 53
B.14. 13 to 14 53
B.15. 14 to 15 54
B.16. 15 to 16 54
B.17. 16 to 17 54
B.18. 17 to 18 54
B.19. 18 to 19 54
B.20. 19 to 20 54
B.21. 20 to 21 55
Acknowledgements 55
Author's Address 55

1. Introduction

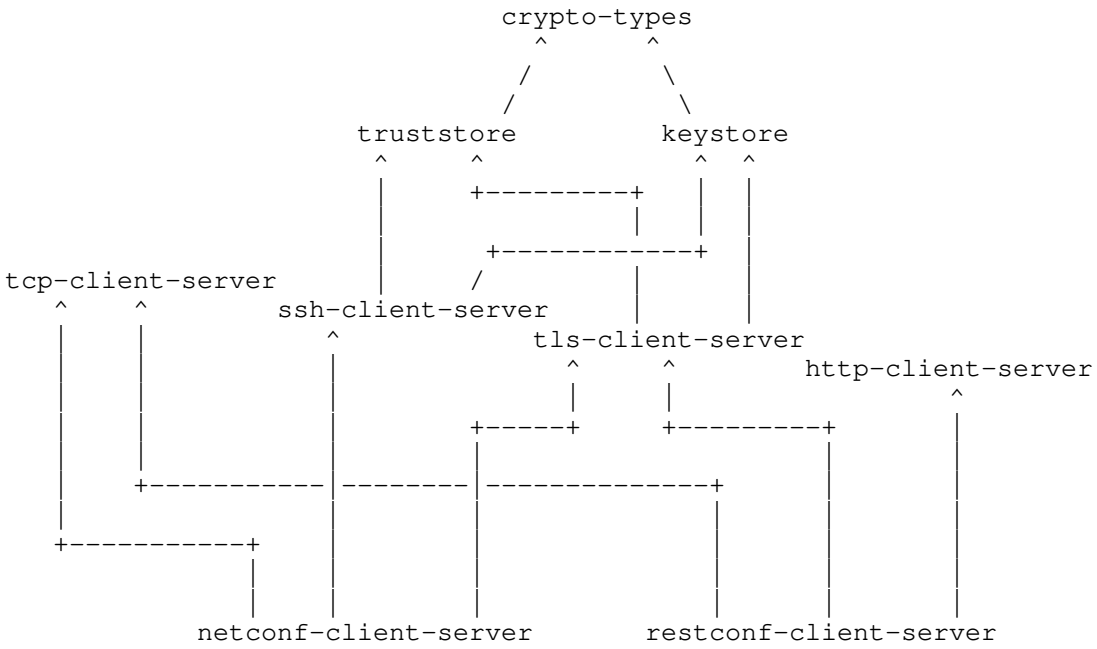
This document defines two YANG [RFC7950] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server [RFC8040]. Both modules support the TLS [RFC8446] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [RFC8071].

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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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

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
```

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

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
```

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

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 tree diagram [RFC8340] 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>
                    <cleartext-password>secret</cleartext-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>
                    <cleartext-password>secret</cleartext-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>
                <cleartext-password>secret</cleartext-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@2020-08-20.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 2020-08-20 {
       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

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
```

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

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
```

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

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 tree diagram [RFC8340] 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\
  ore-reference>
              </ca-certs>
              <ee-certs>
                <truststore-reference>trusted-client-ee-certs</truststo\
  ore-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@2020-08-20.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 2020-08-20 {
      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 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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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.

B.21. 20 to 21

* Cleaned up titles in the IANA Consideratons section

* Fixed issues found by the SecDir review of the "keystore" 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, 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: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-22

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:

* "2020-08-20" --> 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 3
  1.1. Relation to other RFCs 4
  1.2. Specification Language 6
  1.3. Adherence to the NMDA 6
2. The "ietf-ssh-common" Module 6
  2.1. Data Model Overview 6
  2.2. Example Usage 9
  2.3. YANG Module 9
3. The "ietf-ssh-client" Module 19
  3.1. Data Model Overview 19
  3.2. Example Usage 22
  3.3. YANG Module 26
4. The "ietf-ssh-server" Module 33
  4.1. Data Model Overview 33
  4.2. Example Usage 36

4.3. YANG Module 40
5. Security Considerations 49
5.1. The "ietf-ssh-common" YANG Module 49
5.2. The "ietf-ssh-client" YANG Module 50
5.3. The "ietf-ssh-server" YANG Module 51
6. IANA Considerations 52
6.1. The "IETF XML" Registry 52
6.2. The "YANG Module Names" Registry 52
7. References 53
7.1. Normative References 53
7.2. Informative References 54
Appendix A. Change Log 56
A.1. 00 to 01 56
A.2. 01 to 02 56
A.3. 02 to 03 56
A.4. 03 to 04 57
A.5. 04 to 05 57
A.6. 05 to 06 57
A.7. 06 to 07 57
A.8. 07 to 08 58
A.9. 08 to 09 58
A.10. 09 to 10 58
A.11. 10 to 11 58
A.12. 11 to 12 58
A.13. 12 to 13 59
A.14. 13 to 14 59
A.15. 14 to 15 59
A.16. 15 to 16 59
A.17. 16 to 17 59
A.18. 17 to 18 60
A.19. 18 to 19 60
A.20. 19 to 20 61
A.21. 20 to 21 61
A.22. 21 to 22 61
Acknowledgements 61
Author's Address 61

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.

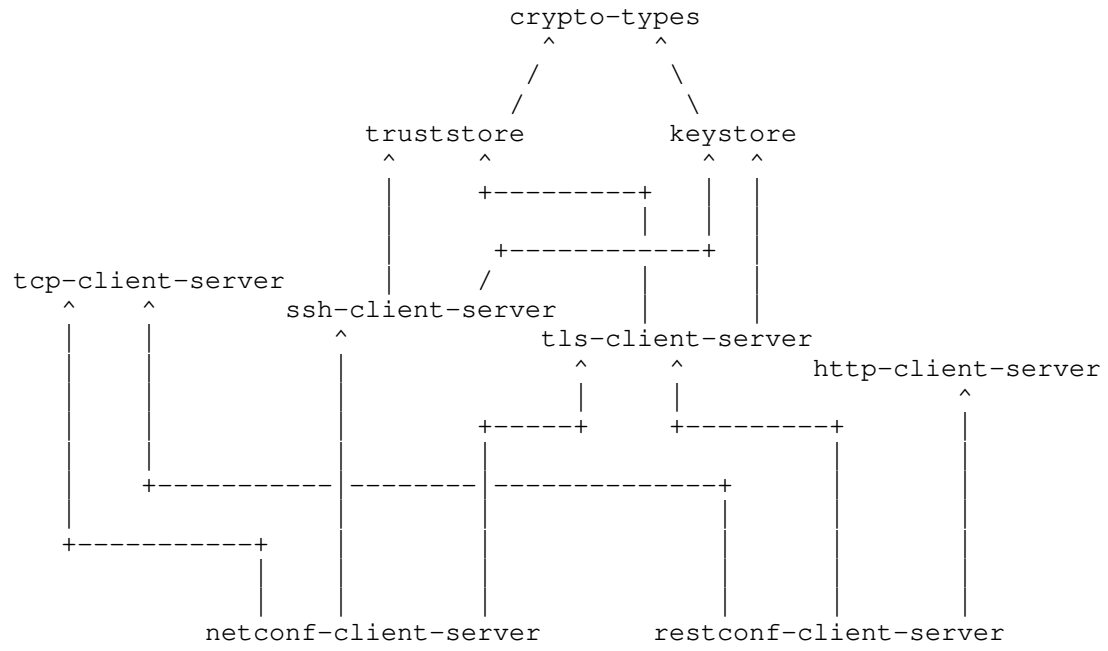
The modules defined in this document optionally support [RFC6187] enabling X.509v3 certificate based host keys and public keys.

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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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

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-ecc
     +-- ssh-x509-certs
     +-- ssh-dh-group-exchange
     +-- ssh-ctr
     +-- ssh-sha2
```

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

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
```

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

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

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

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@2020-08-20.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 2020-08-20 {
       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

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 illustatrating 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-transport-params-config
  +-- ssh-client-keepalives
  +-- client-identity-password
  +-- client-identity-publickey
  +-- client-identity-hostbased
  +-- client-identity-none
```

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

3.1.2. Groupings

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

```
Groupings:
  +-- ssh-client-grouping
```

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

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! {client-identity-password}?
       |  |  +---u ct:password-grouping
       |  +-- 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>
           <public-key>
             <name>corp-fw1</name>
             <public-key-format>ct:ssh-public-key-format</public-key-fo\
   rmat>
             <public-key>base64encodedvalue==</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>
           </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>
         <truststore-reference>trusted-ssh-public-keys</truststore-refe\
   rence>
       </ssh-host-keys>
       <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>
     </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@2020-08-20.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 2020-08-20; // 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 2020-08-20 {
        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"';
              }
            }
          }
          container password {
            if-feature client-identity-password;
            presence
              "Indicates that password-based authenitcation is
               configured.";
            description
              "A password to be used to authenicate the client's
               identity.";
            uses ct:password-grouping;
          }
          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

This section defines a YANG 1.1 [RFC7950] module called "ietf-ssh-server". A high-level overview of the module is provided in Section 4.1. Examples illustatrating 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-transport-params-config
  +-- ssh-server-keepalives
  +-- client-auth-config-supported
  +-- client-auth-publickey
  +-- client-auth-password
  +-- client-auth-hostbased
  +-- client-auth-none
```

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

4.1.2. Groupings

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

```
Groupings:
  +-- ssh-server-grouping
```

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

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@2020-08-20.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 2020-08-20; // 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 2020-08-20 {
      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., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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.

A.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.

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, Balazs Kovacs, Benoit Claise, Bert Wijnen, David Lamparter, Gary Wu, Juergen Schoenwaelder, Ladislav Lhotka, Liang Xia, Martin Bjorklund, Mehmet Ersue, Michal Vasko, Phil Shafer, Radek Krejci, Sean Turner, Tom Petch.

Special acknowledgement goes to Gary Wu who contributed the "ietf-ssh-common" module.

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: 5 April 2021

K. Watsen
Watsen Networks
R. Housley
Vigil Security, LLC
S. Turner
sn3rd

2 October 2020

Conveying a Certificate Signing Request (CSR) in a Secure Zero Touch Provisioning (SZTP) Bootstrapping Request
draft-ietf-netconf-sztp-csr-00

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:

* "2020-10-02" --> 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

This Internet-Draft will expire on 5 April 2021.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Overview 3
  1.2. Terminology 3
  1.3. Requirements Language 4
2. The "ietf-sztp-csr" Module 4
  2.1. Data Model Overview 4
  2.2. Example Usage 7
  2.3. YANG Module 13
3. Security Considerations 23

3.1. SZTP-Client Considerations 23
3.1.1. Ensuring the Integrity of Asymmetric Private Keys . . 23
3.1.2. Reuse of a Manufacturer-generated Private Key 23
3.1.3. Replay Attack Protection 24
3.1.4. Connecting to an Untrusted Bootstrap Server 24
3.1.5. Selecting the Best Origin Authentication Mechanism . 25
3.1.6. Clearing the Private Key and Associated Certificate 25
3.2. SZTP-Server Considerations 25
3.2.1. Conveying Proof of Possession to a CA 25
3.2.2. Supporting SZTP-Clients that don't trust the SZTP-Server 25
3.2.3. YANG Module Considerations 26
4. IANA Considerations 26
4.1. The "IETF XML" Registry 26
4.2. The "YANG Module Names" Registry 26
5. References 27
5.1. Normative References 27
5.2. Informative References 28
Authors' Addresses 29

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 7.1 of [RFC8040].

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 RFC 8792 ================

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?    ietf-crypto-types:csr-info
```

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?    ct:csr-info
```

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 RFC 8792 ================

   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-algorithm": {
                "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 RFC 8792 ================

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 RFC 8792 ================

{
  "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-data": "base64encodedvalue=="
              },
              {
                "name": "Newly-Generated LDevID Cert",
                "cert-data": "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 RFC 8792 ================

   {
     "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-data": "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-data": "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@2020-10-02.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: YANG Data Types and Groupings 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 2020-10-02 {
      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 IDevID 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:
                   YANG Data Types and Groupings 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 ct:csr-info;
        description
          "A CertificationRequestInfo structure, as defined in
           RFC 2986.

           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
           RFC AAAA:
             YANG Data Types and Groupings for Cryptography";
      }
    }
  }
```

<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 FIXME: guy says that the the keys/certs aren't always stored in the TPM (see private email from Aug 13th) 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., CMC, 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., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-18, 20 August 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>.

[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>.

[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>.

[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-20, 20 August 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-20>.

[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-13, 20 August 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-13>.

[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: 21 February 2021

K. Watsen
Watsen Networks
M. Scharf
Hochschule Esslingen

20 August 2020

YANG Groupings for TCP Clients and TCP Servers
draft-ietf-netconf-tcp-client-server-08

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:

* "2020-08-20" --> 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 21 February 2021.

Table of Contents

1. Introduction 3
  1.1. Relation to other RFCs 3
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
2. The "ietf-tcp-common" Module 5
  2.1. Data Model Overview 5
  2.2. Example Usage 8
  2.3. YANG Module 8
3. The "ietf-tcp-client" Module 11
  3.1. Data Model Overview 11
  3.2. Example Usage 13
  3.3. YANG Module 14
4. The "ietf-tcp-server" Module 21
  4.1. Data Model Overview 21
  4.2. Example Usage 22
  4.3. YANG Module 23
5. Security Considerations 25
  5.1. The "ietf-tcp-common" YANG Module 25
  5.2. The "ietf-tcp-client" YANG Module 26
  5.3. The "ietf-tcp-server" YANG Module 27
6. IANA Considerations 27
  6.1. The "IETF XML" Registry 27
  6.2. The "YANG Module Names" Registry 28
7. References 28
  7.1. Normative References 28

7.2. Informative References 29
Appendix A. Change Log 31
A.1. 00 to 01 31
A.2. 01 to 02 31
A.3. 02 to 03 31
A.4. 03 to 04 31
A.5. 04 to 05 31
A.6. 05 to 06 31
A.7. 06 to 07 32
A.8. 08 to 09 32
Authors' Addresses 32

1. Introduction

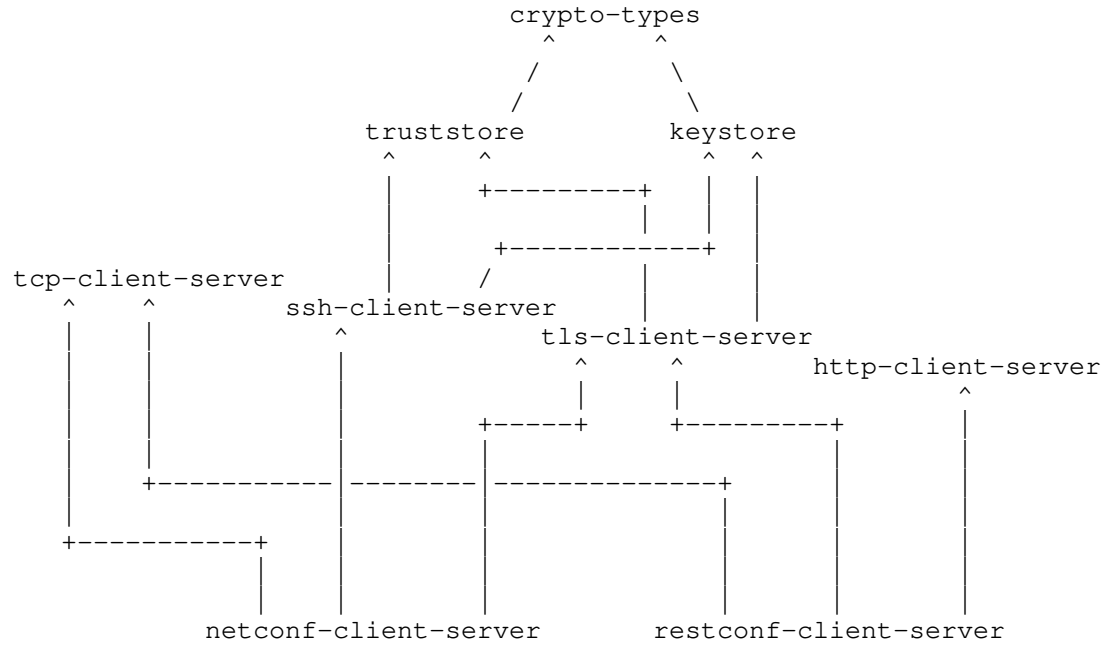
This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers, either as standalone or in conjunction with a stack protocol layer specific configurations.

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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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

This section defines a YANG 1.1 [RFC7950] module called "ietf-tcp-common". A high-level overview of the module is provided in Section 2.1. Examples illustatrating 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 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
```

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

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
```

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

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@2020-08-20.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 2020-08-20 {
    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

This section defines a YANG 1.1 [RFC7950] module called "ietf-tcp-client". A high-level overview of the module is provided in Section 3.1. Examples illustatrating 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
  +-- socks5-gss-api
  +-- socks5-username-password
```

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

3.1.2. Groupings

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

```
Groupings:
  +-- tcp-client-grouping
```

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

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
    |                       +---u ct:password-grouping
    +---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 two examples showing the "tcp-client-grouping" populated with some data. This example shows a TCP-client configured to not connect via a proxy:

```
<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>
```

This example shows a TCP-client configured to connect via a proxy:

```
<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>
  <proxy-server>
    <socks5-parameters>
      <remote-address>proxy.my-domain.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>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@2020-08-20.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:   <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 2020-08-20 {
      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;
                       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-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

This section defines a YANG 1.1 [RFC7950] module called "ietf-tcp-server". A high-level overview of the module is provided in Section 4.1. Examples illustatrating 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 following diagram lists all the "grouping" statements defined in the "ietf-tcp-server" module:

```
Groupings:
  +-- tcp-server-grouping
```

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

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@2020-08-20.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 2020-08-20 {
      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-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-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., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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.

A.8. 08 to 09

* 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.

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: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-22

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:

* "2020-08-20" --> 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 3
  1.1. Relation to other RFCs 4
  1.2. Specification Language 6
  1.3. Adherence to the NMDA 6
2. The "ietf-tls-common" Module 6
  2.1. Data Model Overview 7
  2.2. Example Usage 10
  2.3. YANG Module 10
3. The "ietf-tls-client" Module 19
  3.1. Data Model Overview 19
  3.2. Example Usage 21
  3.3. YANG Module 25
4. The "ietf-tls-server" Module 32
  4.1. Data Model Overview 32
  4.2. Example Usage 35

4.3. YANG Module 39
5. Security Considerations 46
5.1. The "ietf-tls-common" YANG Module 46
5.2. The "ietf-tls-client" YANG Module 47
5.3. The "ietf-tls-server" YANG Module 48
6. IANA Considerations 48
6.1. The "IETF XML" Registry 48
6.2. The "YANG Module Names" Registry 49
7. References 49
7.1. Normative References 49
7.2. Informative References 51
Appendix A. Change Log 53
A.1. 00 to 01 53
A.2. 01 to 02 53
A.3. 02 to 03 53
A.4. 03 to 04 53
A.5. 04 to 05 54
A.6. 05 to 06 54
A.7. 06 to 07 54
A.8. 07 to 08 54
A.9. 08 to 09 54
A.10. 09 to 10 55
A.11. 10 to 11 55
A.12. 11 to 12 55
A.13. 12 to 13 55
A.14. 12 to 13 56
A.15. 13 to 14 56
A.16. 14 to 15 56
A.17. 15 to 16 56
A.18. 16 to 17 56
A.19. 17 to 18 57
A.20. 18 to 19 57
A.21. 19 to 20 57
A.22. 20 to 21 58
A.23. 21 to 22 58
Acknowledgements 58
Author's Address 58

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.

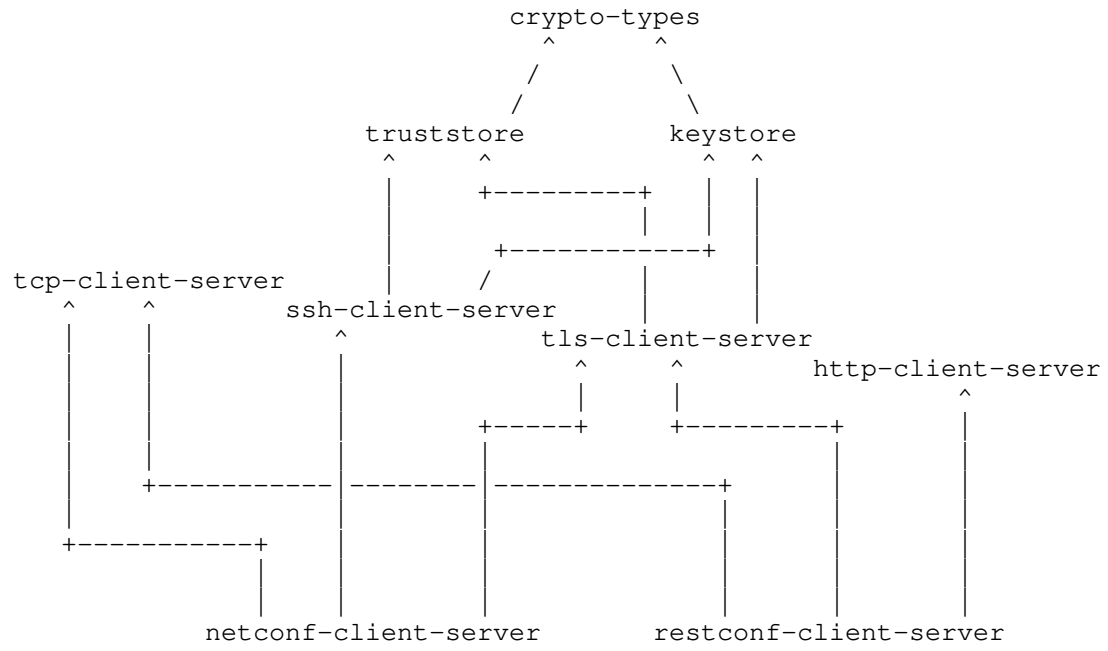
The client and server YANG modules in this document each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy 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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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

This section provides an overview of the "ietf-tls-common" module in terms of its features, identitiesm and groupings.

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
```

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

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
```

> 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 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

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

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@2020-08-20.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 2020-08-20 {
       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

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 illustatrating 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-hello-params-config
  +-- tls-client-keepalives
  +-- x509-certificate-auth
  +-- raw-public-key-auth
  +-- psk-auth
```

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

3.1.2. Groupings

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

```
Groupings:
  +-- tls-client-grouping
```

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

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
    |           +-- id?
    |                    string
    +-- 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?              empty {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</truststo\
   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@2020-08-20.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 2020-08-20; // 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 2020-08-20 {
    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.";
          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).";
              uses ks:local-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)";
              }
            }
          }
        }
      } // 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 {
          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"';
            }
```

```
          }
        }
        leaf psks {
          if-feature "psk-auth";
          type empty;
          description
            "Indicates that the TLS client can authenticate TLS servers
             using configure 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.";
        }
      } // 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

This section defines a YANG 1.1 [RFC7950] module called "ietf-tls-server". A high-level overview of the module is provided in Section 4.1. Examples illustatrating 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-hello-params-config
  +-- tls-server-keepalives
  +-- client-auth-config-supported
  +-- x509-certificate-auth
  +-- raw-public-key-auth
  +-- psk-auth
```

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

4.1.2. Groupings

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

```
Groupings:
  +-- tls-server-grouping
```

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

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
       |           +-- id_hint?
       |                       string
       +-- 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?              empty {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@2020-08-20.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 2020-08-20; // 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 2020-08-20 {
        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).";
              uses ks:local-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)";
              }
            }
          }
        }
      } // 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 {
          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"';
              }
            }
          }
          leaf psks {
            if-feature "psk-auth";
            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.";
          }
        } // 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., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

[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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020,
<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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.

A.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.

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, Balazs Kovacs, Benoit Claise, Bert Wijnen, David Lamparter, Gary Wu, Henk Birkholz, Juergen Schoenwaelder, Ladislav Lhotka, Liang Xia, Martin Bjorklund, Mehmet Ersue, Michal Vasko, Phil Shafer, Radek Krejci, Sean Turner, and Tom Petch.

Special acknowledgement goes to Gary Wu who contributed the "ietf-tls-common" module.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 February 2021

K. Watsen
Watsen Networks

20 August 2020

A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-13

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:

* "2020-08-20" --> 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 3
  1.1. Relation to other RFCs 3
  1.2. Specification Language 5
  1.3. Adherence to the NMDA 5
2. The "ietf-truststore" Module 5
  2.1. Data Model Overview 5
  2.2. Example Usage 12
  2.3. YANG Module 21
3. Support for Built-in Trust Anchors 29
4. Security Considerations 32
  4.1. Data at Rest 32
  4.2. The "ietf-truststore" YANG Module 33
5. IANA Considerations 33
  5.1. The "IETF XML" Registry 33
  5.2. The "YANG Module Names" Registry 33
6. References 34
  6.1. Normative References 34
  6.2. Informative References 34
Appendix A. Change Log 36
  A.1. 00 to 01 36
  A.2. 01 to 02 36
  A.3. 02 to 03 36
  A.4. 03 to 04 37
  A.5. 04 to 05 37
  A.6. 05 to 06 37

A.7. 06 to 07 37
A.8. 07 to 08 37
A.9. 08 to 09 38
A.10. 09 to 10 38
A.11. 10 to 11 38
A.12. 11 to 12 38
A.13. 12 to 13 39
Acknowledgements 39
Author's Address 39

1. Introduction

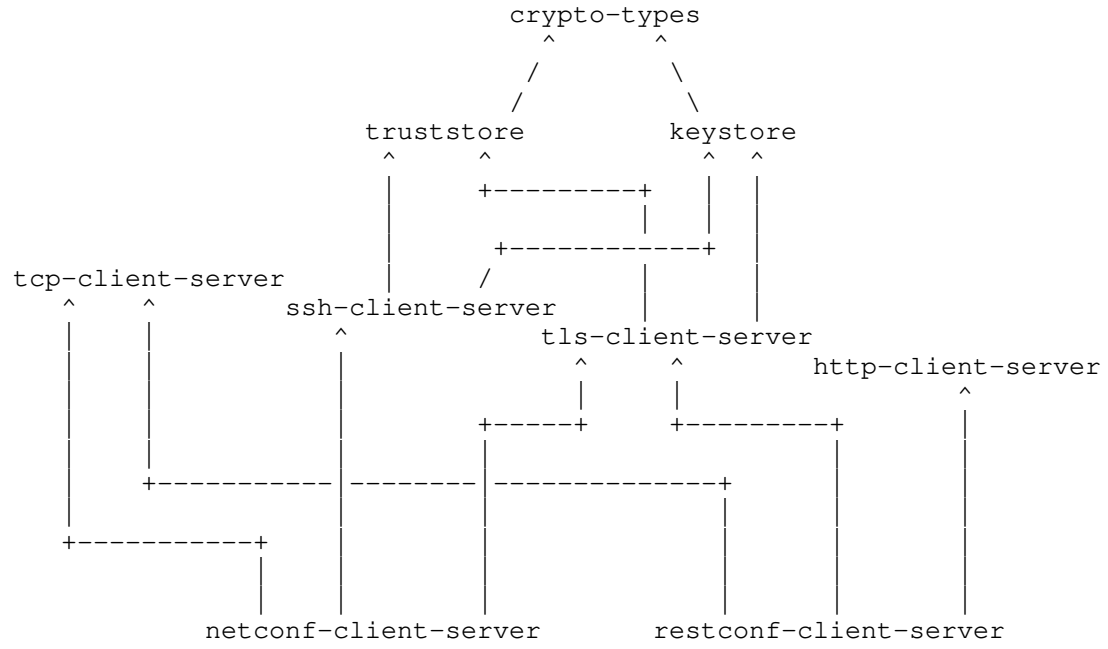
This document defines a YANG 1.1 [RFC7950] data model for configuring globally-accessible bags of certificates and public keys that can be referenced by other data models for trust.

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 normative dependency 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. Hyperlinks to each RFC are provided below the diagram.



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

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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.

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

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

2.1.2. Typedefs

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

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

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

Comments:

* All 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
```

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

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 whether a bag of certificates can be defined locally or as a reference to a bag in the truststore.

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

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

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

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

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

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

Comments:

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

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

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

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

2.1.3.3. The "truststore-grouping" Grouping

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

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

Comments:

* The "truststore-grouping" grouping 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 2.1.4.7 of [I-D.ietf-netconf-crypto-types]. Thus the "cert-data" node is a CMS structure that can be composed of a chain of one or more certificates. Additionally, the "certificate-expiration" notification enables the server to alert clients when certificates are nearing or have already expired.

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

2.1.4. Protocol-accessible Nodes

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

```
module: ietf-truststore
  +--rw truststore
     +---u truststore-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
  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
  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
```

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
        +--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

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

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

Comments:

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

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

* The 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 2.1.4.6 of [I-D.ietf-netconf-crypto-types]) for a certificate configured in the truststore in Section 2.2.1.

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

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

2.2.3. The "Local or Truststore" Groupings

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

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

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

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

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

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

```
module: ex-truststore-usage
  +--rw truststore-usage
     +--rw cert* [name]
     |  +--rw name                           string
     |  +--rw (local-or-truststore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw certificate* [name]
     |     |        +--rw name                      string
     |     |        +--rw cert-data
     |     |        |       trust-anchor-cert-cms
     |     |        +---n certificate-expiration
     |     |                {certificate-expiration-notification}?
     |     |           +-- expiration-date    yang:date-and-time
     |     +--:(truststore) {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>
```

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

```
  module ex-truststore-usage {
    yang-version 1.1;

    namespace "http://example.com/ns/example-truststore-usage";
    prefix "etu";

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

    organization
     "Example Corporation";

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

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

    revision "2020-08-20" {
      description
       "Initial version";
      reference
        "RFC BBBB: A YANG Data Model for a Truststore";
    }

    container truststore-usage {
      description
```

```
        "An illustration of the various truststore groupings.";

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

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

2.3. YANG Module

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

<CODE BEGINS> file "ietf-truststore@2020-08-20.yang"

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

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

```
  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: 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 2020-08-20 {
    description
      "Initial version";
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  /****************/
  /*   Features   */
```

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

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

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

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

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

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

     typedef certificate-bag-ref {
       type leafref {
         path "/ts:truststore/ts:certificate-bags/"
              + "ts:certificate-bag/ts:name";
       }
       description
         "This typedef defines a reference to a certificate bag
          defined in the Truststore.";
     }

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

```
          in a certificate bag defined in the Truststore.  This
          typedef requires that there exist a sibling 'leaf' node
          called 'certificate-bag' that SHOULD have the typedef
          'certificate-bag-ref'.";
      }

      typedef public-key-bag-ref {
        type leafref {
          path "/ts:truststore/ts:public-key-bags/"
               + "ts:public-key-bag/ts:name";
        }
        description
          "This typedef define a reference to a public key bag
           defined in the Truststore.";
      }

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


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

      grouping local-or-truststore-certs-grouping {
        description
          "A grouping that allows the certificates to be either
           configured locally, within the using data model, or be a
           reference to a certificate bag stored in the Truststore.";
        choice local-or-truststore {
          nacm:default-deny-write;
          mandatory true;
          description
            "A choice between an inlined definition and a definition
             that exists in the Truststore.";
          case local {
            if-feature "local-definitions-supported";
```

```
          container local-definition {
            description
              "A container for locally configured trust anchor
               certificates.";
            list certificate {
              key "name";
              min-elements 1;
              description
                "A trust anchor certificate.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this certificate.";
              }
              uses ct:trust-anchor-cert-grouping {
                refine "cert-data" {
                  mandatory true;
                }
              }
            }
          }
        }
        case truststore {
          if-feature "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.";
          }
        }
      }
    }


    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;
        description
          "A choice between an inlined definition and a definition
           that exists in the Truststore.";
        case local {
```

```
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "A container to hold local public key definitions.";
            list public-key {
              key name;
              description
                "A public key definition.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this public key.";
              }
              uses ct:public-key-grouping;
            }
          }
        }
        case truststore {
          if-feature "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.";
          }
        }
      }
    }

    grouping truststore-grouping {
      description
        "A grouping definition that 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 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-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

[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-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

[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-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

[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-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

[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-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

[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-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

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

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

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

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

Appendix A. Change Log

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

A.1. 00 to 01

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

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

A.2. 01 to 02

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

A.3. 02 to 03

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

A.4. 03 to 04

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

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

A.5. 04 to 05

* Renamed "trust-anchors" to "truststore"

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

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

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

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

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

A.6. 05 to 06

* Editorial changes only.

A.7. 06 to 07

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

* Added PSKs and raw public keys to Truststore.

A.8. 07 to 08

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

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

* Removed PSK from model.

A.9. 08 to 09

* Removed remaining PSK references from text.
* Wrapped each top-level list with a container.
* Introduced "bag" term.
* Merged "SSH Public Keys" and "Raw Public Keys" in a single "Public Keys" bag. Consuming downstream modules (i.e., "ietf-[ssh/tls]-[client/server]) refine the "public-key-format" to be either SSH or TLS specific as needed.

A.10. 09 to 10

* Removed "algorithm" node from examples.
* Removed the no longer used statements supporting the old "ssh-public-key" and "raw-public-key" nodes.
* Added a "Note to Reviewers" note to first page.

A.11. 10 to 11

* Corrected module prefix registered in the IANA Considerations section.
* Modified 'local-or-truststore-certs-grouping' to use a list (not a leaf-list).
* Added new example section "The Local or Truststore Groupings".
* Clarified expected behavior for "built-in" certificates in <operational>
* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
* Updated the Security Considerations section.

A.12. 11 to 12

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

A.13. 12 to 13

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

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
Internet-Draft
Intended status: Standards Track
Expires: 6 May 2021

G. Zheng
T. Zhou
Huawei
T. Graf
Swisscom
P. Francois
INSA-Lyon
P. Lucente
NTT

2 November 2020

UDP-based Transport for Configured Subscriptions
draft-ietf-netconf-udp-notif-01

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 6 May 2021.

Table of Contents

1. Introduction 2
2. Configured Subscription to UDP-Notif 4
3. UDP-Based Transport 4
3.1. Design Overview 4
3.2. Format of the UDP-Notif Message Header 5
3.3. Options 6
3.3.1. Segmentation Option 7
3.4. Data Encoding 8
4. Applicability 8
4.1. Congestion Control 8
4.2. Message Size 9
4.3. Reliability 9
5. A YANG Data Model for Management of UDP-Notif 9
6. YANG Module 10
7. IANA Considerations 12
8. Security Considerations 13
9. Acknowledgements 13
10. References 13
10.1. Normative References 13
10.2. Informative References 15
Authors' Addresses 15

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 their 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.ietf-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.ietf-netconf-distributed-notif].

Section 2 describes the control of the proposed transport mechanism. Section 3 details the notification mechanism and message format. Section 4.1 discusses congestion control. Section 4 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 segmentation 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, CBOR [RFC7049], JSON, and XML. [I-D.ietf-netconf-notification-messages] describes the structure of the Notification Message for single notifications and bundled notifications.

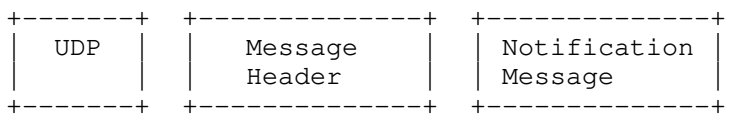


Figure 1: UDP-Notif Message Overview

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.

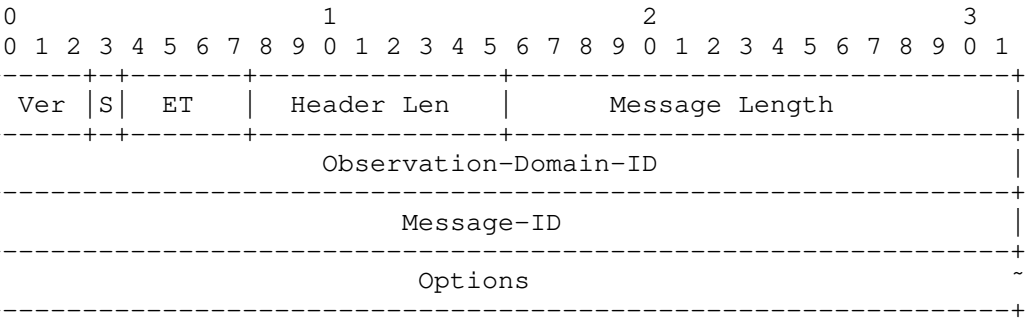


Figure 2: UDP-Notif Message Header Format

The Message Header contains the following field:

* Ver represents the PDU (Protocol Data Unit) encoding version. The initial version value is 0.

* S represents the space of encoding type specified in the ET field. When S is unset, ET represents the standard encoding types as defined in this document. When S is set, ET represents a private space to be freely used for non standard encodings.

* ET is a 4 bit identifier to indicate the encoding type used for the Notification Message. 16 types of encoding can be expressed. When the S bit is unset, the following values apply:

  - 0: CBOR;

  - 1: JSON;

  - 2: XML;

- others are reserved.

* Header Len 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.

* Observation-Domain-ID is a 32-bit identifier of the Observation Domain that led to the production of the notification message, as defined in [I-D.ietf-netconf-notification-messages]. 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.

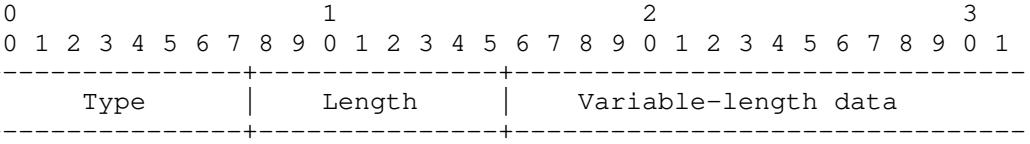


Figure 3: Generic Option Format

* Type: 1 octet describing the option type;

* Length: 1 octet representing the total number of octets in the TLV, including the Type and Length fields;

* Variable-length data: 0 or more octets of TLV Value.

3.3.1. Segmentation Option

The UDP payload length is limited to 65535. Application level headers will make the actual payload shorter. Even though binary encodings such as 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-segment-size option to control the maximum size of a payload.

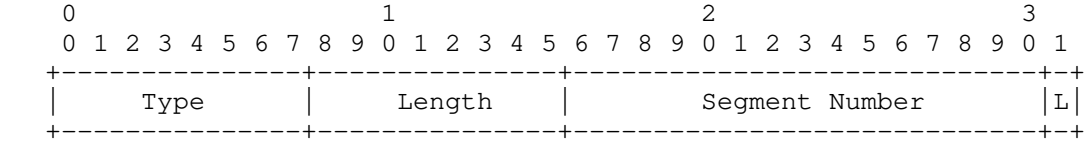


Figure 4: Segmentation Option Format

The Segmentation Option is to be included when the message content is segmented into multiple pieces. Different segments of one message share the same Message ID. An illustration is provided in Figure 4. The fields of this TLV are:

* Type: Generic option field which indicates a Segmentation Option. The Type value is to be asigned.

* Length: Generic option field which indicates the length of this option. It is a fixed value of 4 octets for the Segmentation Option.

* Segment Number: 15-bit value indicating the sequence number of the current segment. The first segment of a segmented message has a Segment Number value of 0.

* L: is a flag to indicate whether the current segment is the last one of the message. When 0 is set, the current segment is not the last one. When 1 is set, the current segment is the last one, meaning that the total number of segments used to transport this message is the value of the current Segment Number + 1.

An implementation of this specification MUST NOT rely on IP fragmentation by default to carry large messages. An implementation of this specification MUST either restrict the size of individual messages carried over this protocol, or support the segmentation option.

3.4. Data Encoding

UDP-Notif message data can be encoded in 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. Applicability

In this section, we provide an applicability statement for the proposed mechanism, following the recommendations of [RFC8085].

The proposed mechanism falls in the category of UDP applications "designed for use within the network of a single network operator or on networks of an adjacent set of cooperating network operators, to be deployed in controlled environments". Implementations of the proposed mechanism should thus follow the recommendations in place for such specific applications. In the following, we discuss recommendations on congestion control, message size guildelines, reliability considerations.

4.1. Congestion Control

The proposed application falls into the category of applications performing transfer of large amounts of data. It is expected that the operator using the solution configures QoS on its related flows. As per [RFC8085], such applications MAY choose not to implement any form of congestion control, but follow the following principles.

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.

Implementation of the proposal SHOULD NOT push unlimited amounts of traffic by default, and SHOULD require the users to explicitely configure such a mode of operation.

Burst mitigation through packet pacing is RECOMMENDED. Disabling burst mitigation SHOULD require the users to explicitely configure such a mode of operation.

Applications SHOULD monitor packet losses and provide means to the user for retrieving information on such losses. 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.

4.2. Message Size

[RFC8085] recommends not to rely on IP fragmentation for messages whose size result in IP packets exceeding the MTU along the path. The segmentation option of the current specification permits to perform segmentation of the UDP Notif message content so as to not have to rely on IP fragmentation. Implementation of the current specification SHOULD allow for the configuration of the MTU.

4.3. Reliability

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, as information collected for the devices through the means of the proposed mechanism is to be often refreshed.

A collector implementation for this protocol SHOULD deal with potential loss of packets carrying a part of segmented payload, by discarding packets that were actually received, but cannot be re-assembled as a complete message within a given amount of time. This time SHOULD be configurable.

5. 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
```

6. 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";
    }

    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>
```

7. 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

8. Security Considerations

TBD

9. 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.

10. References

10.1. 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>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[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>.

10.2. Informative References

[I-D.ietf-netconf-distributed-notif]
Zhou, T., Zheng, G., Voit, E., Graf, T., and P. Francois, "Subscription to Distributed Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-distributed-notif-01, June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-distributed-notif-01>.

[I-D.ietf-netconf-https-notif]
Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for Configured Subscriptions", Work in Progress, Internet-Draft, draft-ietf-netconf-https-notif-04, 27 July 2020, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-https-notif-04.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, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-notification-messages-08.txt>.

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
CH- 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
Netherlands

Email: paolo@ntt.net

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: 6 May 2021

J. Lindblad
Cisco Systems

2 November 2020

Transaction ID Mechanism for NETCONF
draft-lindblad-netconf-transaction-id-00

Abstract

TODO Abstract

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/janlindblad/netconf-transaction-id.

Table of Contents

1. Introduction 2
2. Conventions and Definitions 3
3. NETCONF Extension 3
4. Configuration Retreival 4
4.1. Configuration Response Pruning 5
5. Configuration Update 7
5.1. Conditional Configuration Update 10
6. YANG Modules 13
7. Security Considerations 16
8. IANA Considerations 16
9. Normative References 16
Acknowledgments 16
Author's Address 16

1. Introduction

When a NETCONF client connects with a NETCONF server, a frequently occurring use case is for the client to find out if the configuration has changed since it was last connected. Such changes could occur for example if another NETCONF client has made changes, or another system or operator made changes through other means than NETCONF.

One way of detecting a change for a client would be to retrieve the entire configuration from the server, then compare the result with a previously stored copy at the client side. This approach is not popular with most NETCONF users, however, since it would often be very expensive in terms of communications and computation cost.

Furthermore, even if the configuration is reported to be unchanged, that will not guarantee that the configuration remains unchanged when a client sends a subsequent change request, which arrives soon thereafter.

Evidence of a transaction-id feature being demanded by clients is that several server implementors have built proprietary and mutually incompatible mechanisms for obtaining a transaction id from a NETCONF server.

RESTCONF, RFC 8040 RFC8040 (https://tools.ietf.org/html/rfc8040), defines a mechanism for detecting changes in configuration subtrees based on Entity-tags (ETags). In conjunction with this, RESTCONF provides a way to make configuration changes conditional on the server confiuguration being untouched by others. This mechanism leverages RFC 7232 RFC7232 (https://tools.ietf.org/html/rfc7232) "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests".

This document defines similar functionality for NETCONF, RFC 6241 RFC6241 (https://tools.ietf.org/html/rfc6241).

2. Conventions and Definitions

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.

3. NETCONF Extension

This document describes a NETCONF extension which modifies the behavior of get-config, get-data, edit-config and edit-data such that clients are able to conditionally retrieve and update the configuration in a NETCONF server. NETCONF servers that supports this extension MUST announce the capability "FIXME".

The extended operations defined in this document pertains to YANG containers and list elements. It is NOT REQUIRED that a conforming server allows the extended operations to apply to all containers and list elements in the server configuration. The set of containers and list elements that the server supports with respect to this NETCONF extension are collectively referred to as the "versioned elements".

The NETCONF server will maintain a record of the transaction that last changed each versioned element. This transaction-id meta level data is communicated between the server and client in the form of an XML attribute called "entag". The values for the entag attribute is up to the clients and servers to decice as opaque quantities. It is essential that the entag values have a large value space in order to not run out or collide. They SHOULD be at least 32-bit quantities.

Entag attribute values are encoded as YANG strings.

> Comment, to be removed
>
> Do we want to limit the entag attribute strings in some way? E.g. only base64 characters, some min or max length, ...?

4. Configuration Retreival

When a NETCONF client retrieves the configuration from a NETCONF server that implement this specification, it may request that the configuration is entity tagged. The entity tags are XML attributes added to some of the retrieved configuration elements by the server. These elements are collectively called the "versioned elements".

The entity-tag (entag) attributes are guaranteed to change every time there has been a configuration change at or below the element bearing the attribute.

Clients request entity tags to be added by setting the ietf-netconf-transaction-id:entag attribute to the value "?" on one or more elements in the request. Entags MUST be returned for all descendant versioned elements. In order to request that entags are returned for the entire configuration, the client can place the attribute on the top edit-config or edit-data tags. For more specific retrieval, the client inserts entag attributes in the filter section.

To retrieve entag attributes across the entire NETCONF server configuration, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:ietf-netconf-transaction-id=
       "FIXME">
  <get-config ietf-netconf-transaction-id:entag="?"/>
</rpc>
```

To retrieve entag attributes for "ietf-interfaces", but not for "nacm", a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:ietf-netconf-transaction-id=
       "FIXME">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        ietf-netconf-transaction-id:entag="?"/>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
    </filter>
  </get-config>
</rpc>
```

When a NETCONF server receives a get-config or get-data request containing ietf-netconf-transaction-id:entag attributes with the value "?", it MUST return entag attributes for all versioned elements below this point included in the reply.

The server's response to request above might look like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="def88884321">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface ietf-netconf-transaction-id:entag="abc12345678">
        <name>GigabitEthernet-0/1</name>
        <description>Upward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
      <groups>
        <group>
          <name>admin</name>
          <user-name>sakura</user-name>
          <user-name>joe</user-name>
        </group>
      </groups>
    </nacm>
  </data>
</rpc>
```

4.1. Configuration Response Pruning

A NETCONF client that already knows some entag values may request that the configuration retrieval request is pruned with respect to the client's prior knowledge.

By specifying the previously received entag attribute values in the get-config or get-data request, the client indicates that child elements of already known parts of the configuration SHALL be omitted.

To retrieve only changes for "ietf-interfaces" since the last known transaction-id "abc12345678", but include the entire configuration for "nacm", a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:ietf-netconf-transaction-id=
       "FIXME">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        ietf-netconf-transaction-id:entag="abc12345678"/>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
    </filter>
  </get-config>
</rpc>
```

When a NETCONF server receives a get-config or get-data request containing ietf-netconf-transaction-id:entag attributes with the same value as the entag value known by the server for that element, it MUST prune the contents of that subtree.

In case the element with a matching entag value is a container, the container tag is returned with an entag attribute value of "=". No child elements are returned for the container.

In case the element with a matching entag value is a list element, the list element tag is returned with an entag attribute value of "=". The list element will include the list elemenet keys, but no other child elements.

For example, assuming the NETCONF server configuration is the same as in the previous rpc-reply example, the server's response to request above might look like:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="def88884321">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="def88884321">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface ietf-netconf-transaction-id:entag="=">
        <name>GigabitEthernet-0/1</name>
      </interface>
    </interfaces>
    <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
      <groups>
        <group>
          <name>admin</name>
          <user-name>sakura</user-name>
          <user-name>joe</user-name>
        </group>
      </groups>
    </nacm>
  </data>
</rpc>
```

5. Configuration Update

When a NETCONF client sends an edit-config or edit-data request to a NETCONF server that implements this specification, the client MAY specify a transaction-id value.

If specified, the server MUST use this value as the new value for all entag attribute values of any versioned element touched by the transaction, if and only if the operation is successful. The entag value must be updated regardless of whether an actual value change took place or not. An element is touched if it is mentioned in the transaction, even if it merely sets the element to the same value it already has.

If the server side configuration changes for any reason, and there is no transaction-id value specified by a client, servers that supports this specification MUST update the entag values as if a NETCONF client had made the change and specified a transaction-id. In this case, the server MUST choose a random transaction-id value to use.

Comment, to be removed

Is talk about "random" good enough, or do we need to get specific?

Every time a versioned element has its entag value updated, the same value must be set to all parent versioned elements' entag attributes, cascading all the way to the datastore root.

For example, if a client wishes to update the interface description for interface "GigabitEthernet-0/1" to "Downward Interface", under transaction-id "ghi55550101", it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:ietf-netconf-transaction-id=
       "FIXME">
  <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <ietf-netconf-transaction-id:transaction-id>
      ghi55550101
    </transaction-id>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet-0/1</name>
          <description>Downward Interface</description>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

A subsequent get-config request for "ietf-interfaces", with ietf-netconf-transaction-id:entag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="ghi55550101">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
      <interface ietf-netconf-transaction-id:entag="ghi55550101">
        <name>GigabitEthernet-0/1</name>
        <description>Downward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

In case the server received a configuration change from another source, such as a CLI operator, adding an MTU value for the interface "GigabitEthernet-0/0", a subsequent get-config request for "ietf-interfaces", with ietf-netconf-transaction-id:entag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="auto22223333">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="auto22223333">
      <interface ietf-netconf-transaction-id:entag="auto22223333">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <mtu>768</mtu>
      </interface>
      <interface ietf-netconf-transaction-id:entag="ghi55550101">
        <name>GigabitEthernet-0/1</name>
        <description>Downward Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

5.1. Conditional Configuration Update

Conditional Transactions are useful when a client is interested to make a configuration change, being sure that the server configuration has not changed since the client last inspected it.

By supplying the latest entag values known to the client in its change requests (edit-config etc.), it can request the server to reject the transaction in case any changes have occurred at the server that the client is not yet aware of.

Even if a client is constantly connected to a device, and even possibly receiving notifications when a server device's configuration changes, there is always a possibility that a change is introduced by another party in the time window between when the client last received an update about the server's configuration until the server executes a configuration change request.

By leveraging conditional transactions, this race condition can be eliminated efficiently. If the client provides the transaction-id it expects the device to have as part of it's configuration change request, and the device guarantees to only execute the request in case the transaction-id in the request matches that on the server, the race condition is removed.

When a NETCONF client sends an edit-config or edit-data request to a NETCONF server that implements this specification, the client MAY specify expected entag values on the versioned elements touched by the transaction.

If such an entag value differs from the entag value stored on the server, the server MUST reject the transaction.

If the server rejects the transaction because the configuration entag value differs from the client's expectation, ther server MUST return an rpc-error with the following values:

```
error-tag:      operation-failed
error-type:     protocol
error-severity: error
```

Additionally, the error-info tag MUST contain a sx:structure entag-value-mismatch-error-info, with mismatch-path set to the instance identifier value identifying one of the versioned elements that had an entag value mismatch, and mismatch-entag-value set to the server's current value of the entag attribute for that versioned element.

For example, if a client wishes to delete the interface "GigabitEthernet-0/1" if and only if its configuration has not been altered since this client last synchronized its configuration with the server (at which point it received a transaction-id "ghi55550101"), regardless of any possible changes to other interfaces, it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ietf-netconf-transaction-id=
       "FIXME">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface nc:operation="delete"
                   ietf-netconf-transaction-id:entag="ghi55550101">
          <name>GigabitEthernet-0/1</name>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

If interface "GigabitEthernet-0/1" has the entag value "ghi55550101", as expected by the client, the transaction goes through.

A subsequent get-config request for "ietf-interfaces", with ietf-netconf-transaction-id:entag="?" might then return:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ietf-netconf-transaction-id=
             "FIXME">
  <data ietf-netconf-transaction-id:entag="auto77775511">
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                ietf-netconf-transaction-id:entag="auto77775511">
      <interface ietf-netconf-transaction-id:entag="def88884321">
        <name>GigabitEthernet-0/0</name>
        <description>Management Interface</description>
        <type>ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
      </interface>
    </interfaces>
  </data>
</rpc>
```

If interface "GigabitEthernet-0/1" did not have the entag value "ghi55550101", the server rejects the transaction, and might send:

```
   <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
              xmlns:ietf-netconf-transaction-id=
                "FIXME">
              message-id="1">
     <rpc-error>
       <error-type>protocol</error-type>
       <error-tag>operation-failed</error-tag>
       <error-severity>error</error-severity>
       <error-info>
         <ietf-netconf-transaction-id:entag-value-mismatch-error-info>
           <ietf-netconf-transaction-id:mismatch-path>
             /if:interfaces/if:interface[if:name="GigabitEthernet-0/0"]
           </ietf-netconf-transaction-id:mismatch-path>
           <ietf-netconf-transaction-id:mismatch-entag-value>
             auto77775511
           </ietf-netconf-transaction-id:mismatch-entag-value>
         </ietf-netconf-transaction-id:entag-value-mismatch-error-info>
       </error-info>
     </rpc-error>
   </rpc-reply>
```

Comment, to be removed

In order to reach the full flexibility with the above transaction rejection mechanism, it might make sense to reference parts of the configuration just to see that they have not moved, with no intent to make changes there. To support this use case, a new operation mode "nocreate" might be useful. This would allow an edit config to talk about parts of the configuration which are expected to exist with a particular confiuguration, and to abort the transaction if they do not exist.

Comment, to be removed

NETCONF clients may be equally interested to apply a mechanism similar to entags when retrieving operational state as well, since there is often vey much of this data, and some if changes rather rarely. To support this use case, some sort of server maintained change indicators may be invented, and combined with a similar retrieval filter.

6. YANG Modules

Comment, to be removed

This is YANG 1.1. Do we also want 1.0? Makes it possible to implement on 1.0 servers

```
   module ietf-netconf-transaction-id {
     yang-version 1.1;
     namespace
       'urn:ietf:params:xml:ns:yang:ietf-netconf-transaction-id';
     prefix ietf-netconf-transaction-id;

     import ietf-netconf {
       prefix nc;
     }

     import ietf-netconf-nmda {
       prefix ncds;
     }

     import ietf-yang-structure-ext {
       prefix sx;
     }

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

     contact
       "WG Web:   <http://tools.ietf.org/wg/netconf/>
        WG List:  <netconf@ietf.org>

        Author:   Jan Lindblad
                  <mailto:jlindbla@cisco.com>";

     description
       "NETCONF Transaction ID aware operations for NMDA.

        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 2020-10-01 {
       description
         "Initial revision";
       reference
```

```
        "RFC XXXX: Xxxxxxxxx";
    }

    typedef transaction-id-t {
      type string;
      description
        "Unique value representing a specific transaction";
    }

    grouping transaction-id-grouping {
      leaf transaction-id {
        type transaction-id-t;
        description
          "Transaction-id value selected by the client.  This string
           should be chosen to give a high probability to be unique on
           the server.";
      }
      description
        "Grouping for transaction-id, to be augmented into rpcs
         that modify configuration data stores.";
    }

    augment /nc:edit-config/nc:input {
      uses transaction-id-grouping;
      description
        "Injects the transaction-id leaf into the edit-config
        operation";
    }

    augment /ncds:edit-data/ncds:input {
      uses transaction-id-grouping;
      description
        "Injects the transaction-id leaf into the edit-data
        operation";
    }

    sx:structure entag-value-mismatch-error-info {
      container entag-value-mismatch-error-info {
        description
           "This error is returned by a NETCONF server when a client
            sends a configuration change request, with the additonal
            condition that the server aborts the transaction if the
            server's configuration has changed from what the client
            expects, and the configuration is found not to actually
            not match the client's expectation.";
        leaf mismatch-path {
          type instance-identifier;
          description
```

```
            "Indicates the YANG path to the element with a mismatching
             entag value.";
        }
        leaf mismatch-entag-value {
          type transaction-id-t;
          description
            "Indicates server's value of the entag attribute for one
             mismatching element.";
        }
      }
    }
  }
```

7. Security Considerations

TODO Security

8. IANA Considerations

This document has no IANA actions.

9. 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>.

[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>.

Acknowledgments

TODO acknowledge.

Author's Address

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.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 2
1.1. Terminology 2
2. Data Export capability 3
2.1. Tree Diagram 4
3. YANG Module 4
4. IANA Considerations 11
4.1. Updates to the IETF XML Registry 11
4.2. Updates to the YANG Module Names Registry 11
5. Security Considerations 12
6. Contributors 13
7. References 13
7.1. Normative References 13
7.2. Informative References 14
Appendix A. Usage Example of interaction between telemetry data export capabilities and Adaptive Subscription . . . 14
Authors' Addresses 17

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: 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 2
1.1. Terminology 3
2. Model Overview 4
2.1. Subscription Configuration 5
2.2. YANG RPC 6
2.2.1. "establish-subscription" RPC 6
2.2.2. "modify-subscription" RPC 6
2.3. Notifications for Adaptive Subscribed Content 6
3. Adaptive Subscription YANG Module 7
4. IANA Considerations 12
4.1. Updates to the IETF XML Registry 12
4.2. Updates to the YANG Module Names Registry 12
5. Security Considerations 12
6. Contributors 13
7. References 13
7.1. Normative References 13
7.2. Informative References 14
Appendix A. Example YANG Module 14
A.1. "example-wifi-mac" YANG Module 15
Appendix B. Adaptive Subscription and Notification Example . . . 18
B.1. "edit-config" Example 18
B.2. Create Adaptive Subscription Example 19
B.3. "adaptive-update" notification example 21
Authors' Addresses 22

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