

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

YANG Data Types and Groupings for Cryptography
draft-ietf-netconf-crypto-types-19

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:

* "2021-02-10" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	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	18
2.3. YANG Module	27
3. Security Considerations	47
3.1. No Support for CRMF	48
3.2. No Support for Key Generation	48
3.3. Unconstrained Public Key Usage	48
3.4. Unconstrained Private Key Usage	48
3.5. Strength of Keys Configured	49
3.6. Encrypting Passwords	49
3.7. Deletion of Cleartext Key Values	49
3.8. The "ietf-crypto-types" YANG Module	49
4. IANA Considerations	51
4.1. The "IETF XML" Registry	51
4.2. The "YANG Module Names" Registry	51
5. References	51
5.1. Normative References	51
5.2. Informative References	53
Appendix A. Change Log	55
A.1. I-D to 00	55

A.2.	00 to 01	55
A.3.	01 to 02	56
A.4.	02 to 03	56
A.5.	03 to 04	57
A.6.	04 to 05	57
A.7.	05 to 06	57
A.8.	06 to 07	58
A.9.	07 to 08	58
A.10.	08 to 09	58
A.11.	09 to 10	58
A.12.	10 to 11	59
A.13.	11 to 12	59
A.14.	12 to 13	59
A.15.	13 to 14	59
A.16.	14 to 15	60
A.17.	15 to 16	60
A.18.	16 to 17	60
A.19.	17 to 18	61
A.20.	18 to 19	61
Acknowledgements		61
Author's Address		61

1. Introduction

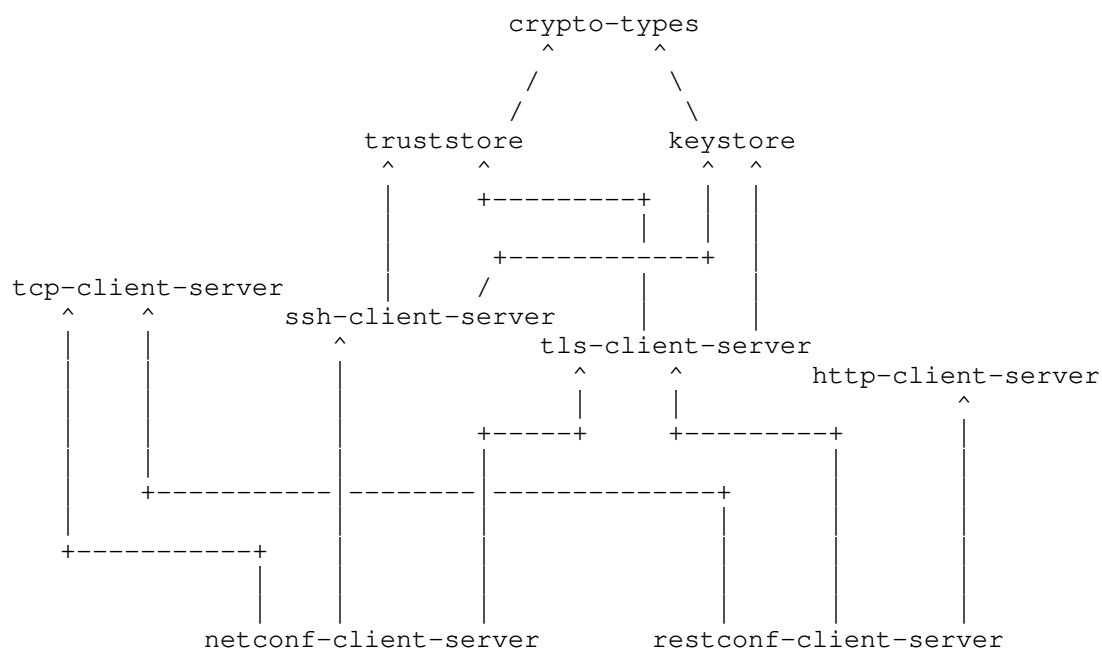
This document presents a YANG 1.1 [RFC7950] module defining identities, typedefs, and groupings useful to cryptographic applications.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the 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 is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

2. The "ietf-crypto-types" Module

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

2.1. Data Model Overview

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

2.1.1. Features

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

Features:

- +-- one-symmetric-key-format
- +-- one-asymmetric-key-format
- +-- certificate-signing-request-generation
- +-- certificate-expiration-notification
- +-- symmetrically-encrypted-value-format
- +-- asymmetrically-encrypted-value-format
- +-- cms-encrypted-data-format
- +-- cms-enveloped-data-format

| 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}?
+-- symmetric-key-format
|   +-- octet-string-key-format
|   +-- one-symmetric-key-format
|       {one-symmetric-key-format}?
+-- encrypted-value-format
|   +-- symmetrically-encrypted-value-format
|       |   {symmetrically-encrypted-value-format}?
|       +-- cms-encrypted-data-format
|           {cms-encrypted-data-format}?
+-- asymmetrically-encrypted-value-format
|   |   {asymmetrically-encrypted-value-format}?
+-- cms-enveloped-data-format
    {cms-enveloped-data-format}?

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

```

Comments:

- * The diagram shows that there are four base identities. The first three identities are used to indicate the format that key data, while the fourth identity is used to indicate the format for encrypted values. The base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of formats, rather than a specific format.
- * The various "leaf" identities define specific encoding formats. The derived identities defined in this document are sufficient for the effort described in Section 1.1 but, by nature of them being identities, additional derived identities MAY be defined by future efforts.

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

2.1.3. Typedefs

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

Typedefs:

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

| The diagram above uses syntax that is similar to but not
| 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 distinguish themselves by the expected nature of their content. These typedefs are defined to facilitate common modeling needs.

2.1.4. Groupings

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

```
* encrypted-value-grouping
* password-grouping
* symmetric-key-grouping
* public-key-grouping
* asymmetric-key-pair-grouping
* trust-anchor-cert-grouping
* end-entity-cert-grouping
* generate-csr-grouping
* asymmetric-key-pair-with-cert-grouping
* asymmetric-key-pair-with-certs-grouping
```

Each of these groupings are presented in the following subsections.

2.1.4.1. The "encrypted-value-grouping" Grouping

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

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

Comments:

- * The "encrypted-by" node is an empty container (difficult to see in the diagram) that a consuming module MUST augment key references into. The "ietf-crypto-types" module is unable to populate this container as the module only defines groupings. Section 2.2.1 presents an example illustrating a consuming module populating the "encrypted-by" container.
- * The "encrypted-value" node is the value, encrypted by the key referenced by the "encrypted-by" node, and encoded in the format appropriate for the kind of key it was encrypted by.
 - If the value is encrypted by a symmetric key, then the encrypted value is encoded using the format associated with the "symmetrically-encrypted-value-format" identity.
 - If the value is encrypted by an asymmetric key, then the encrypted value is encoded using the format associated with the "asymmetrically-encrypted-value-format" identity.

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

2.1.4.2. The "password-grouping" Grouping

This section presents 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-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-format  identityref
        +-- encrypted-value        binary
```

Comments:

- * For the referenced grouping statement(s):
 - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.
- * The "choice" statement enables the password data to be cleartext or encrypted, as follows:
 - The "cleartext-password" node can encode any cleartext value.
 - The "encrypted-password" node's structure is discussed in Section 2.1.4.1.

2.1.4.3. The "symmetric-key-grouping" Grouping

This section presents 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-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-format  identityref
        +-- encrypted-value        binary

```

Comments:

- * For the referenced grouping statement(s):
 - The "encrypted-value-grouping" grouping is discussed in Section 2.1.4.1.
- * The "key-format" node is an identity-reference to the "symmetric-key-format" abstract base identity discussed in Section 2.1.2, enabling the symmetric key to be encoded using the format defined by any of the derived identities.
- * The "choice" statement enables the private key data to be cleartext, encrypted, or hidden, as follows:
 - The "cleartext-key" node can encode any cleartext 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 unnecessary 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-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-format  identityref
          +-- 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-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 cleartext, encrypted, or hidden, as follows:
 - The "cleartext-private-key" node can encode any cleartext key value.
 - The "hidden-private-key" node is of type "empty" as the real value cannot be presented via the management interface.
 - The "encrypted-private-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.6. The "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-format    identityref
    |   | +-- 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-format    identityref
|               +-- 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), the "asymmetric-key-pair-with-certs-grouping" (Section 2.1.4.11), and the "password-grouping" (Section 2.1.4.2) grouping statements.

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

```

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 "2021-02-10" {
    description
      "Initial version";
    reference

```

```

    "RFC AAAA: Common YANG Data Types for Cryptography";
}

```

```

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

```

```

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

```

```

    }
    description
        "An asymmetric key pair with associated certificates.";
    }
}

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

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 "/ecty:asymmetric-keys/ecty:asymmetric-key/"
          + "ecty: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-format identityref
|   |   |   |   |   +--rw encrypted-value binary
|   +--rw asymmetric-keys
|   |   +--rw asymmetric-key* [name]
|   |   |   +--rw name string
|   |   |   +--rw public-key-format identityref
|   |   |   +--rw public-key binary
|   |   |   +--rw private-key-format? identityref
|   |   |   +--rw (private-key-type)
|   |   |   |   +--:(cleartext-private-key)
|   |   |   |   |   +--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-format  identityref
        +---rw encrypted-value        binary
+---rw certificates
  +---rw certificate* [name]
    +---rw name                      string
    +---rw cert-data                  end-entity-cert-cms
    +---n certificate-expiration
      {certificate-expiration-notification}?
      +--- expiration-date            yang:date-and-time
+---x generate-certificate-signing-request
  {certificate-signing-request-generation}?
  +---w input
  | +---w csr-info                    ct:csr-info
  +---ro output
    +---ro certificate-signing-request ct:csr
+---rw passwords
  +---rw password* [name]
    +---rw name                      string
    +---rw (password-type)
      +---:(cleartext-password)
      | +---rw cleartext-password?    string
      +---:(encrypted-password) {password-encryption}?
      +---rw encrypted-password
        +---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-format identityref
            +---rw encrypted-value        binary

```

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

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

```

<symmetric-keys
  xmlns="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:one-symmetric-key-format</key-format>
    <encrypted-key>
      <encrypted-by>
        <asymmetric-key-ref>ex-hidden-asymmetric-key</asymmetric-key\
- ref>
      </encrypted-by>
      <encrypted-value-format>
        ct:cms-enveloped-data-format
      </encrypted-value-format>
      <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-rsa-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-asymmetric-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-rsa-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>
  <encrypted-private-key>
    <encrypted-by>
      <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
    </encrypted-by>

```



```

        <encrypted-value-format>
          ct:cms-encrypted-data-format
        </encrypted-value-format>
        <encrypted-value>base64encodedvalue==</encrypted-value>
      </encrypted-private-key>
    </asymmetric-key>
  </asymmetric-keys>

  <passwords
    xmlns="http://example.com/ns/example-crypto-types-usage"
    xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <password>
      <name>ex-cleartext-password</name>
      <cleartext-password>super-secret</cleartext-password>
    </password>
    <password>
      <name>ex-encrypted-password</name>
      <encrypted-password>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
        </encrypted-by>
        <encrypted-value-format>
          ct:cms-encrypted-data-format
        </encrypted-value-format>
        <encrypted-value>base64encodedvalue==</encrypted-value>
      </encrypted-password>
    </password>
  </passwords>

```

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

The following example illustrates the "generate-certificate-signing-request" action, discussed in Section 2.1.4.9, 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], [RFC7093], [RFC8174], [RFC8341], and [ITU.X690.2015].

<CODE BEGINS> file "ietf-crypto-types@2021-02-10.yang"

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

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

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

organization

"IETF NETCONF (Network Configuration) Working Group";

contact

"WG Web: <<http://datatracker.ietf.org/wg/netconf/>>

WG List: <<mailto:netconf@ietf.org>>

Author: Kent Watsen <<mailto:kent+ietf@watsen.net>>;

description

"This module defines common YANG types for cryptographic applications.

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC AAAAA (<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 2021-02-10 {

description

"Initial version";

reference

"RFC AAAAA: YANG Data Types and Groupings for Cryptography";

}

/*****/

/* Features */

/*****/

feature one-symmetric-key-format {

description

"Indicates that the server supports the

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

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

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

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

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

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

    feature 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 encryption
             of symmetric keys.";
    }

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

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

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

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

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

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

    identity rsa-private-key-format {
        base "private-key-format";
        description
            "Indicates that the private key value is encoded
             as an RSAPrivateKey (from RFC 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).";
  }

  /*****
  /* 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.

    How the associated algorithm is known is outside the
    scope of this module. This statement also applies when
    the octet string has been encrypted.";
}

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

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

```



```

/*****/

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

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

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

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

identity cms-enveloped-data-format {
  if-feature "cms-enveloped-data-format";
  base "asymmetrically-encrypted-value-format";
  description
    "Indicates that the encrypted value conforms to the
    'enveloped-data-cms' type with the following constraints:

    The EnvelopedData structure MUST have exactly one
    'RecipientInfo'."

```

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

Otherwise, if the asymmetric key supports key agreement (e.g., ECC), then the 'RecipientInfo' must be a 'KeyAgreeRecipientInfo'. The 'OriginatorIdentifierOrKey' value must use the 'OriginatorPublicKey' alternative. The 'UserKeyingMaterial' value must not be present. There must be exactly one 'RecipientEncryptedKeys' value having the 'KeyAgreeRecipientIdentifier' set to 'rKeyId' with the value set using 'method 1' in RFC 7093 over the recipient's public key.";

reference

"RFC 5652: Cryptographic Message Syntax (CMS)

RFC 7093:

Additional Methods for Generating Key
Identifiers Values

ITU-T X.690:

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

}

```

/*****
/*  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-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 a reference to the key that encrypted the value to be augmented in. The referenced key MUST be a symmetric key or an asymmetric key.

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

The leaf nodes MUST be direct descendents in the data tree, and MAY be direct descendents in the schema tree.";

}

leaf encrypted-value-format {

type identityref {

base encrypted-value-format;

}

mandatory true;

description

"Identifies the format of the 'encrypted-value' leaf.

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

If 'encrypted-by' points to an asymmetric key, then an


```

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

grouping password-grouping {
    description
        "A password 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.";
                uses encrypted-value-grouping;
            }
        }
    }
}

grouping symmetric-key-grouping {
    description
        "A symmetric key.";
    leaf key-format {
        nacm:default-deny-write;
    }
}

```

```

type identityref {
  base symmetric-key-format;
}
description
  "Identifies the symmetric key's format. Implementations
  SHOULD ensure that the incoming symmetric key value is
  encoded in the specified format.

  For encrypted keys, the value is the same as it would
  have been if the key were not encrypted.";
}
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-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.

       For encrypted keys, the value is the same as it would have
       been if the key were not encrypted.";
  }
  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-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.";
    list certificate {
      key "name";
      description
        "A certificate for this asymmetric key.";
      leaf name {
        type string;
        description
          "An arbitrary name for the certificate.";
      }
      uses end-entity-cert-grouping {
        refine cert-data {
          mandatory true;
        }
      }
    }
  }
  uses generate-csr-grouping;
} // asymmetric-key-pair-with-certs-grouping
}

<CODE ENDS>

```

3. Security Considerations

3.1. No Support for CRMF

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

3.2. No Support for Key Generation

Early revisions of this document included "rpc" statements for generating symmetric and asymmetric keys. These statements were removed due to an inability to obtain consensus for how to 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. Unconstrained Public Key Usage

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

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

The "asymmetric-key-pair-with-cert-grouping" grouping uses the aforementioned "asymmetric-key-pair-grouping" grouping, whereby each certificate may constrain the usage of the public key according to local policy.

3.4. Unconstrained Private Key Usage

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

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

3.5. 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 than the strength of the underlying transport.

3.6. Encrypting Passwords

The module contained within this document enables passwords to be encrypted. Passwords may be encrypted via a symmetric key using the "cms-encrypted-data-format" format. This format uses the CMS EncryptedData structure, which allows any encryption algorithm to be used.

In order to thwart rainbow attacks, algorithms that result in a unique output for the same input SHOULD be used. For instance, AES using "EBC" SHOULD NOT be used to encrypt passwords, whereas "CBC" mode is okay since it a unique initialization vector (IV) should be used for each run.

3.7. Deletion of Cleartext Key Values

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

The cleartext key values are the "cleartext-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).

3.8. 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 IESG
XML: N/A, the requested URI is an XML namespace.

4.2. The "YANG Module Names" Registry

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

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

5. References

5.1. Normative References

[ITU.X680.2015]

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>>.
- [RFC7093] Turner, S., Kent, S., and J. Manger, "Additional Methods for Generating Key Identifiers Values", RFC 7093, DOI 10.17487/RFC7093, December 2013, <<https://www.rfc-editor.org/info/rfc7093>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

- [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-18, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>>.
- [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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.
- [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-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.
- [I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-

netconf-ssh-client-server-22, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.

[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-08, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.

[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>>.

[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.

A.20. 18 to 19

- * Added a "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.
- * Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.
- * Changed the encryption strategy, after conferring with Russ Housley.
- * Added a "password-grouping" example to the "crypto-types-usage" example.
- * Added an "Encrypting Passwords" section to Security Consideration.
- * Addressed other comments raised by YANG Doctor.

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, Russ Housley, 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 document 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.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	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 overwhelm the route processor resources, it is not uncommon that data records are published directly from processors on line cards to target Receivers to further increase efficiency on the routing system.

This document complements 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,

- o the Subscriber generates the subscription instructions to express what and how the collector want to receive the data;
- o 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 collect metrics according to the Component Subscription, add its metadata, encapsulate and push data to the Receiver where packets are reassembled and decapsulated.

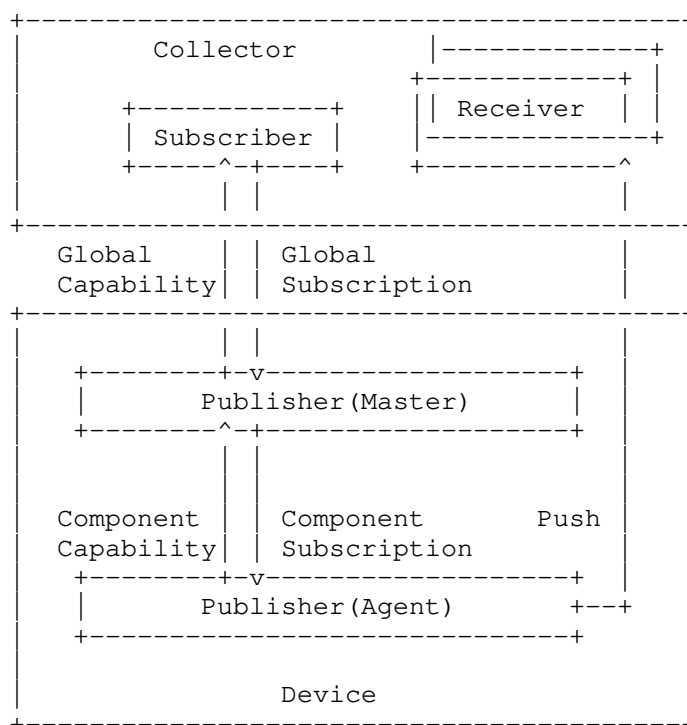


Fig. 2 The Distributed Data Export Framework

Master and Agents interact with each other in several ways:

- o Agents need to register at the Master at the beginning of their process life-cycle
- o Contracts are created between the Master and each Agent on the Component Capability, and the format for streaming data structure.

- o The Master relays the component subscriptions to the Agents.
- o 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:

- o Inherit the Global Subscription properties from Publisher Master for its Component Subscription;
- o share the same life-cycle as the Global Subscription;
- o 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 responsible 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:

- o /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
Futurewei
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.

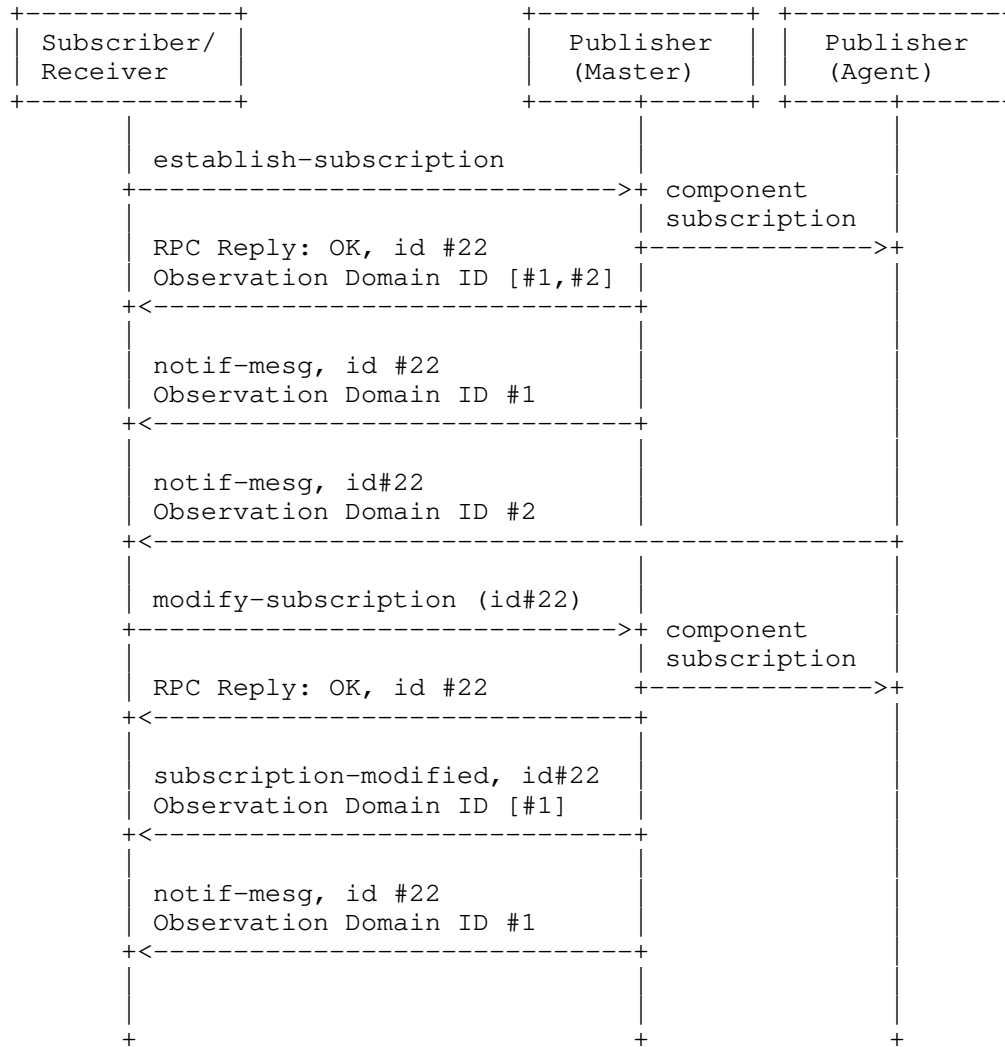


Fig. 3 Call Flow for Dynamic Subscription

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-notifications">
      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.

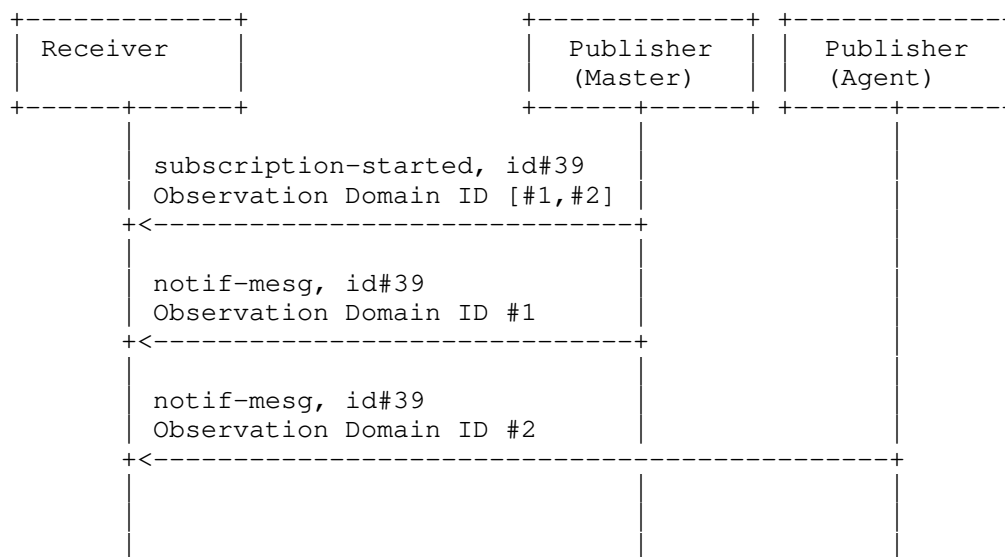


Fig. 8 Call Flow for Configured Subscription

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: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

YANG Groupings for HTTP Clients and HTTP Servers
draft-ietf-netconf-http-client-server-06

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:

* "2021-02-10" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	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	16
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	26
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	29
A.3. 02 to 03	30
A.4. 03 to 04	30
A.5. 04 to 05	30
A.6. 05 to 06	30
Acknowledgements	31
Author's Address	31

1. Introduction

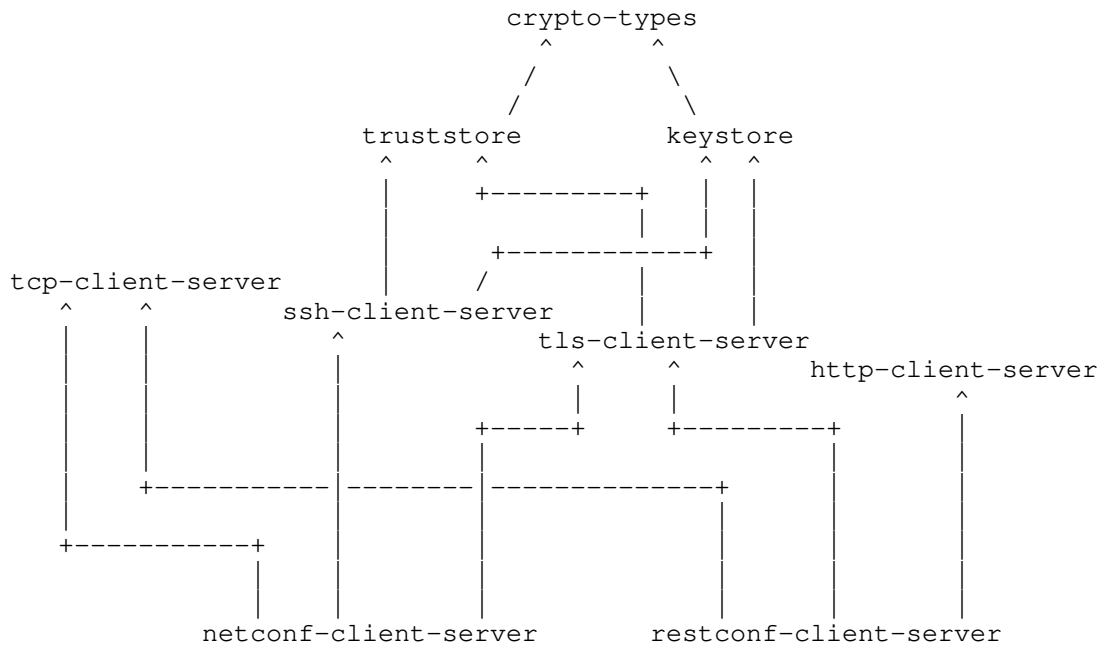
This document defines two YANG 1.1 [RFC7950] modules: the first defines a minimal grouping for configuring an HTTP client, and the second defines a minimal grouping for configuring an HTTP server. It is intended that these groupings will be used to help define the configuration for simple HTTP-based 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, ultimately, enable the configuration of the 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 is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

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 illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-http-client" module in terms of its features and groupings.

2.1.1. Features

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

Features:

```
+-- 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 "ietf-http-client" module defines the following "grouping" statements:

```
* http-client-identity-grouping
* http-client-grouping
* http-client-stack-grouping
```

Each of these groupings are presented in the following subsections.

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

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

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

Comments:

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

- * The "client-identity" node is a "presence" container so 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>
        </keepalives>
      </tcp-client-parameters>
    </https-proxy>
  </proxy-connect>
</http-client>
```

```

        <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@2021-02-10.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 2021-02-10 {
  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.";

```
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 illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-http-server" module in terms of its features and groupings.

3.1.1. Features

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

Features:

```
+-- client-auth-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 "ietf-http-server" module defines the following "grouping" statements:

```
* http-server-grouping
* http-server-stack-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "http-server-grouping" Grouping

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

```
grouping http-server-grouping
+-- server-name?            string
+-- client-authentication! {client-auth-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 be 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@2021-02-10.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 2021-02-10 {
  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 than
    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 IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-http-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

name: ietf-http-client
namespace: urn:ietf:params:xml:ns:yang:ietf-http-client
prefix: httpc
reference: RFC GGGG

name: ietf-http-server
namespace: urn:ietf:params:xml:ns:yang:ietf-http-server
prefix: https
reference: RFC GGGG

6. References

6.1. Normative References

- [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-18, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>>.
- [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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.

[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-netconf-client-server]

Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.

[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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.

[I-D.ietf-netconf-ssh-client-server]

Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.

[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-08, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.

[I-D.ietf-netconf-tls-client-server]

Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.

[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>>.

- [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.

A.6. 05 to 06

- * Removed note questioning if okay for app to augment-in a 'path' node when needed, discussed during the 108 session.
- * Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

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, Rob Wilton (contributor), and Willy Tarreau.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

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

M. Jethanandani
Kloud Services
K. Watsen
Watsen Networks
February 22, 2021

An HTTPS-based Transport for YANG Notifications
draft-ietf-netconf-https-notif-08

Abstract

This document defines a protocol for sending notifications over HTTPS. YANG modules for configuring publishers are also defined. Examples are provided illustrating how to configure various publishers.

This document requires that the publisher is a "server" (e.g., a NETCONF or RESTCONF server), but does not assume that the receiver is a server.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 26, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Applicability Statement	3
1.2. Note to RFC Editor	3
1.3. Abbreviations	4
1.4. Terminology	4
1.4.1. Subscribed Notifications	4
2. Overview of Publisher to Receiver Interaction	4
3. Discovering a Receiver's Capabilities	5
3.1. Applicability	5
3.2. Request	5
3.3. Response	6
3.4. Example	6
4. Sending Event Notifications	7
4.1. Request	7
4.2. Response	8
4.3. Example	8
5. The "ietf-subscribed-notif-receivers" Module	9
5.1. Data Model Overview	9
5.2. YANG Module	9
6. The "ietf-https-notif-transport" Module	12
6.1. Data Model Overview	12
6.2. YANG module	14
7. Security Considerations	17
8. IANA Considerations	18
8.1. The "IETF XML" Registry	18
8.2. The "YANG Module Names" Registry	18
8.3. The "Capabilities for HTTPS Notification Receivers" Registry	18
9. References	20
9.1. Normative references	20
9.2. Informative references	21
Appendix A. Configuration Examples	22
A.1. Using Subscribed Notifications (RFC 8639)	22
A.2. Not Using Subscribed Notifications	24
Acknowledgements	27
Authors' Addresses	27

1. Introduction

This document defines a protocol for sending notifications over HTTPS. Using HTTPS maximizes transport-level interoperability, while allowing for a variety of encoding options. This document defines support for JSON and XML; future efforts may define support for other encodings (e.g., binary).

This document also defines two YANG 1.1 [RFC7950] modules that extend the data model defined in Subscription to YANG Notifications [RFC8639], enabling the configuration of HTTPS-based receivers.

An example module illustrating the configuration of a publisher not using the data model defined in RFC 8639 is also provided.

Configured subscriptions enable a server, acting as a publisher of notifications, to proactively push notifications to external receivers without the receivers needing to first connect to the server, as is the case with dynamic subscriptions.

1.1. Applicability Statement

While the YANG modules have been defined as an augmentation of Subscription to YANG Notifications [RFC8639], the notification method defined in this document MAY be used outside of Subscription to YANG Notifications [RFC8639] by using some of the definitions from this module along with the grouping defined in Groupings for HTTP Clients and Servers [I-D.ietf-netconf-http-client-server]. For an example on how that can be done, see Section A.2.

1.2. Note to RFC Editor

This document uses several placeholder values throughout the document. Please replace them as follows and remove this section before publication.

RFC XXXX, where XXXX is the number assigned to this document at the time of publication.

RFC YYYY, where YYYY is the number assigned to [I-D.ietf-netconf-http-client-server].

2021-02-22 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].

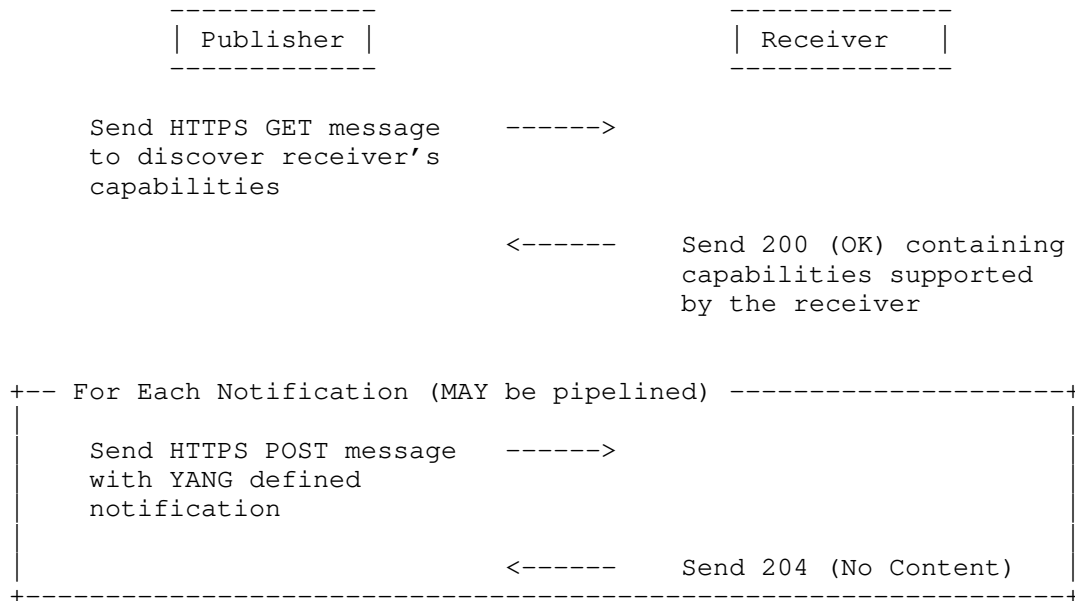
- o Subscribed Notifications

2. Overview of Publisher to Receiver Interaction

The protocol consists of two HTTP-based target resources presented by the receiver. These two resources are sub-paths of a common resource that the publisher must know (e.g. specified in its configuration data model).

- o A target resource enabling the publisher to discover what optional capabilities a receiver supports. Publishers SHOULD query this target before sending any notifications or if ever an error occurs.
- o A target resource enabling the publisher to send one or more notification to a receiver. This document defines support for sending only one notification per message; a future effort MAY extend the protocol to send multiple notifications per message.

The protocol is illustrated in the diagram below:



Note that, for RFC 8639 configured subscriptions, the very first notification must be the "subscription-started" notification.

The POST messages MAY be "pipelined" (not illustrated in the diagram above), whereby multiple notifications are sent without waiting for the HTTP response for a previous POST.

3. Discovering a Receiver's Capabilities

3.1. Applicability

For publishers using Subscription to YANG Notifications [RFC8639], dynamic discovery of a receiver's supported encoding is necessary only when the "/subscriptions/subscription/encoding" leaf is not configured, per the "encoding" leaf's description statement in the "ietf-subscribed-notification" module.

3.2. Request

To learn the capabilities of a receiver, a publisher can issue an HTTPS GET request to the "capabilities" resource under a known path on the receiver with "Accept" header set using the "application/xml" and/or "application/json" media-types, with the latter as mandatory to implement, and the default in case the type is not specified.

3.3. Response

The receiver responds with a "200 (OK)" message, having the "Content-Type" header set to either "application/xml" or "application/json" (which ever was selected), and containing in the response body a list of the receiver's capabilities encoded in the selected format.

Even though a YANG module is not defined for this interaction, the response body MUST conform to the following YANG-modeled format:

```
container receiver-capabilities {
  description
    "A container for a list of capabilities supported by
    the receiver.";
  leaf-list receiver-capability {
    type "inet:uri";
    description
      "A capability supported by the receiver. A full list of
      capabilities is defined in the 'Capabilities for HTTPS
      Notification Receivers' registry (see RFC XXXX).";
  }
}
```

As it is possible that the receiver may return custom capability URIs, the publisher MUST ignore any capabilities that it does not recognize.

3.4. Example

The publisher can send the following request to learn the receiver capabilities. In this example, the "Accept" states that the receiver wants to receive the capabilities response in XML but, if not supported, then in JSON.

```
GET /some/path/capabilities HTTP/1.1
Host: example.com
Accept: application/xml, application/json
```

If the receiver is able to reply using "application/xml", and assuming it is able to receive JSON and XML encoded notifications, and it is able to process the RFC 8639 state machine, the response might look like this:

```
HTTP/1.1 200 OK
Date: Wed, 26 Feb 2020 20:33:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/xml
Content-Length: nnn
```

```
<receiver-capabilities>
  <receiver-capability>\
    urn:ietf:capability:https-notif-receiver:encoding:json\
  </receiver-capability>
  <receiver-capability>\
    urn:ietf:capability:https-notif-receiver:encoding:xml\
  </receiver-capability>
  <receiver-capability>\
    urn:ietf:capability:https-notif-receiver:encoding:rfc8639-enabled\
  </receiver-capability>
</receiver-capabilities>
```

If the receiver is unable to reply using "application/xml", the response might look like this:

```
HTTP/1.1 200 OK
Date: Wed, 26 Feb 2020 20:33:30 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/json
Content-Length: nnn
```

```
{
  receiver-capabilities {
    "receiver-capability": [
      "urn:ietf:capability:https-notif-receiver:encoding:json",
      "urn:ietf:capability:https-notif-receiver:encoding:xml",
      "urn:ietf:capability:https-notif-receiver:encoding:rfc8639-enabled"
    ]
  }
}
```

4. Sending Event Notifications

4.1. Request

The publisher sends an HTTPS POST request to the "relay-notification" resource under a known path on the receiver with the "Content-Type" header set to either "application/json" or "application/xml" and a body containing the notification encoded using the specified format.

XML-encoded notifications are encoded using the format defined by NETCONF Event Notifications [RFC5277] for XML.

JSON-encoded notifications are encoded the same as specified in Section 6.4 in RESTCONF [RFC8040] with the following deviations:

- o The notifications do not contain the "data:" prefix used by SSE.
- o Instead of saying that, for JSON-encoding purposes, the module name for the "notification" element is "ietf-restconf, the module name will instead be "ietf-https-notif".

4.2. Response

The response should be "204 (No Content)".

4.3. Example

An XML-encoded notification might be sent as follows:

```
POST /some/path/relay-notification HTTP/1.1
Host: example.com
Content-Type: application/xml

<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2019-03-22T12:35:00Z</eventTime>
  <event xmlns="https://example.com/example-mod">
    <event-class>fault</event-class>
    <reporting-entity>
      <card>Ethernet0</card>
    </reporting-entity>
    <severity>major</severity>
  </event>
</notification>
```

A JSON-encoded notification might be sent as follows:


```
POST /some/path/relay-notification HTTP/1.1
Host: example.com
Content-Type: application/json
```

```
{
  "ietf-https-notif:notification": {
    "eventTime": "2013-12-21T00:01:00Z",
    "example-mod:event" : {
      "event-class" : "fault",
      "reporting-entity" : { "card" : "Ethernet0" },
      "severity" : "major"
    }
  }
}
```

And, in either case, the response might be as follows:

```
HTTP/1.1 204 No Content
Date: Wed, 26 Feb 2020 20:33:30 GMT
Server: example-server
```

5. The "ietf-subscribed-notif-receivers" Module

5.1. Data Model Overview

This YANG module augments the "ietf-subscribed-notifications" module to define a choice of transport types that other modules such as the "ietf-https-notif-transport" module can use to define a transport specific receiver.

```
module: ietf-subscribed-notif-receivers
  augment /sn:subscriptions:
    +--rw receiver-instances
      +--rw receiver-instance* [name]
        +--rw name      string
        +--rw (transport-type)
      augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
        +--rw receiver-instance-ref?  leafref
```

5.2. YANG Module

The YANG module imports Subscription to YANG Notifications [RFC8639].

```
<CODE BEGINS> file "ietf-subscribed-notif-receivers@2021-02-22.yang"
module ietf-subscribed-notif-receivers {
  yang-version 1.1;
  namespace
```

```
"urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers";  
prefix "snr";
```

```
import ietf-subscribed-notifications {  
  prefix sn;  
  reference  
    "RFC 8639: Subscription to YANG Notifications";  
}
```

```
organization  
  "IETF NETCONF Working Group";
```

```
contact  
  "WG Web: <http://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  
  "This YANG module is implemented by Publishers implementing  
  the 'ietf-subscribed-notifications' module defined in RFC 8639.
```

While this module is defined in RFC XXXX, which primarily defines an HTTPS-based transport for notifications, this module is not HTTP-specific. It is a generic extension that can be used by any 'notif' transport.

This module defines two 'augment' statements. One statement augments a 'container' statement called 'receiver-instances' into the top-level 'subscriptions' container. The other statement, called 'receiver-instance-ref', augments a 'leaf' statement into each 'receiver' that references one of the afore mentioned receiver instances. This indirection enables multiple configured subscriptions to send notifications to the same receiver instance.

Copyright (c) 2021 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 "2021-02-22" {
  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. The 'case' statements must
           be augmented in by other modules.";
      }
      description
        "A list of all receiver instances.";
    }
  }
  description
    "Augment the subscriptions container to define the
     transport type.";
}

augment
  "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
  leaf receiver-instance-ref {
    type leafref {
```

```

        path "/sn:subscriptions/snr:receiver-instances/" +
            "snr:receiver-instance/snr:name";
    }
    description
        "Reference to a receiver instance.";
}
description
    "Augment the subscriptions container to define an optional
    reference to a receiver instance.";
}
}
<CODE ENDS>

```

6. The "ietf-https-notif-transport" Module

6.1. Data Model Overview

This YANG module is a definition of a set of receivers that are interested in the notifications published by the publisher. The module contains the TCP, TLS and HTTPS parameters that are needed to communicate with the receiver. The module augments the "ietf-subscribed-notif-receivers" module to define a transport specific receiver.

As mentioned earlier, it uses a POST method to deliver the notification. The "http-receiver/tls/http-client-parameters/path" leaf defines the path for the resource on the receiver, as defined by "path-absolute" in URI Generic Syntax [RFC3986]. The user-id used by Network Configuration Access Control Model [RFC8341], is that of the receiver and is derived from the certificate presented by the receiver as part of "receiver-identity".

An abridged tree diagram representing the module is shown below.

```

module: ietf-https-notif-transport
  augment /sn:subscriptions/snr:receiver-instances
    /snr:receiver-instance/snr:transport-type:
      +--:(https)
        +--rw https-receiver
          +--rw (transport)
            +--:(tcp) {tcp-supported,not http:tcp-supported}?
              +--rw tcp
                +--rw tcp-client-parameters
                  +--rw remote-address      inet:host
                  +--rw remote-port?       inet:port-number
                  +--rw local-address?     inet:ip-address
                  |
                  +--rw {local-binding-supported}?

```

```

+--rw local-port?          inet:port-number
|   {local-binding-supported}?
+--rw proxy-server! {proxy-connect}?
|   ...
+--rw keepalives!
|   ...
+--rw http-client-parameters
+--rw client-identity!
|   ...
+--rw proxy-connect! {proxy-connect}?
|   ...
+--:(tls) {tls-supported}?
+--rw tls
+--rw tcp-client-parameters
|   +--rw remote-address      inet:host
|   +--rw remote-port?       inet:port-number
|   +--rw local-address?     inet:ip-address
|   |   {local-binding-supported}?
|   +--rw local-port?        inet:port-number
|   |   {local-binding-supported}?
|   +--rw proxy-server! {proxy-connect}?
|   |   ...
|   +--rw keepalives!
|   |   ...
+--rw tls-client-parameters
+--rw client-identity!
|   ...
+--rw server-authentication
|   ...
+--rw hello-params
|   {tls-client-hello-params-config}?
|   ...
+--rw keepalives {tls-client-keepalives}?
|   ...
+--rw http-client-parameters
+--rw client-identity!
|   ...
+--rw proxy-connect! {proxy-connect}?
|   ...
+--rw path                  string
+--rw receiver-identity {receiver-identity}?
+--rw cert-maps
+--rw cert-to-name* [id]
+--rw id                    uint32
+--rw fingerprint          x509c2n:tls-fingerprint
+--rw map-type              identityref
+--rw name                  string

```

6.2. YANG module

The YANG module imports A YANG Data Model for SNMP Configuration [RFC7407], Subscription to YANG Notifications [RFC8639], and YANG Groupings for HTTP Clients and HTTP Servers [I-D.ietf-netconf-http-client-server].

The YANG module is shown below.

```
<CODE BEGINS> file "ietf-https-notif-transport@2021-02-22.yang"
module ietf-https-notif-transport {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-https-notif-transport";
  prefix "hnt";

  import ietf-x509-cert-to-name {
    prefix x509c2n;
    reference
      "RFC 7407: YANG Data Model for SNMP Configuration.";
  }

  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: Subscription to YANG Notifications";
  }

  import ietf-subscribed-notif-receivers {
    prefix snr;
    reference
      "RFC XXXX: An HTTPS-based Transport for
        Configured Subscriptions";
  }

  import ietf-http-client {
    prefix httpc;
    reference
      "RFC YYYY: YANG Groupings for HTTP Clients and HTTP Servers";
  }

  organization
    "IETF NETCONF Working Group";

  contact
    "WG Web:  <http://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
  "This YANG module is implemented by Publishers that implement
  the 'ietf-subscribed-notifications' module defined in RFC 8639.

  This module augments a 'case' statement called 'https' into
  the 'choice' statement called 'transport-type' defined
  by the 'ietf-https-notif-transport' module defined in RFC XXXX.

  Copyright (c) 2021 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 "2021-02-22" {
  description
    "Initial Version.";
  reference
    "RFC XXXX, YANG Data Module for HTTPS Notifications.";
}

feature receiver-identity {
  description
    "Indicates that the server supports filtering notifications
    based on the receiver's identity derived from its TLS
    certificate.";
}

identity https {
  base sn:transport;
  description
    "HTTPS transport for notifications.";
}
```

```
grouping https-receiver-grouping {
  description
    "A grouping that may be used by other modules wishing to
    configure HTTPS-based notifications without using RFC 8639.";
  uses http:http-client-stack-grouping {
    refine "transport/tcp" {
      // create the logical impossibility of enabling the
      // "tcp" transport (i.e., "HTTP" without the 'S').
      if-feature "not http:tcp-supported";
    }
    augment "transport/tls/tls/http-client-parameters" {
      leaf path {
        type string;
        mandatory true;
        description
          "URI prefix to the target resources. Under this
          path the receiver must support both the 'capabilities'
          and 'relay-notification' resource targets, as described
          in RFC XXXX.";
      }
      description
        "Augmentation to add a receiver-specific path for the
        'capabilities' and 'relay-notification' resources.";
    }
  }
  container receiver-identity {
    if-feature receiver-identity;
    description
      "Maps the receiver's TLS certificate to a local identity
      enabling access control to be applied to filter out
      notifications that the receiver may not be authorized
      to view.";
    container cert-maps {
      uses x509c2n:cert-to-name;
      description
        "The cert-maps container is used by a TLS-based HTTP
        server to map the HTTPS client's presented X.509
        certificate to a 'local' username. If no matching and
        valid cert-to-name list entry is found, the publisher
        MUST close the connection, and MUST NOT send any
        notifications over it.";
      reference
        "RFC 7407: A YANG Data Model for SNMP Configuration.";
    }
  }
}

augment "/sn:subscriptions/snr:receiver-instances/" +
```



```
        "snr:receiver-instance/snr:transport-type" {
    case https {
        container https-receiver {
            description
                "The HTTPS receiver to send notifications to.";
            uses https-receiver-grouping;
        }
    }
    description
        "Augment the transport-type choice to include the 'https'
        transport.";
    }
}
}
<CODE ENDS>
```

7. Security Considerations

The YANG 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.

The YANG module in this document makes use of grouping that are defined in YANG Groupings for HTTP Clients and HTTP Servers [I-D.ietf-netconf-http-client-server], and A YANG Data Model for SNMP Configuration [RFC7407]. Please see the Security Considerations section of those documents for considerations related to sensitivity and vulnerability of the data nodes defined in them.

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:

- o The "path" node in "ietf-subscribed-notif-receivers" module can be modified by a malicious user to point to an invalid URI.

Some of the readable data nodes in YANG module may be considered sensitive or vulnerable in some network environments. It is thus

important to control read access (e.g., via get, get-config, or notification) to these data nodes. The model does not define any readable subtrees and data nodes.

Some of the RPC operations in YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. The model does not define any RPC operations.

8. IANA Considerations

8.1. The "IETF XML" Registry

This document registers two URIs in the "ns" subregistry of the "IETF XML" registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-https-notif-transport
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

8.2. The "YANG Module Names" Registry

This document registers two YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

name: iETF-subscribed-notif-receivers
namespace: urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers
prefix: snr
reference: RFC XXXX

name: iETF-https-notif-transport
namespace: urn:ietf:params:xml:ns:yang:ietf-https-notif-transport
prefix: hnt
reference: RFC XXXX

8.3. The "Capabilities for HTTPS Notification Receivers" Registry

Following the guidelines defined in [RFC8126], this document defines a new registry called "Capabilities for HTTPS Notification Receivers". This registry defines capabilities that can be supported by HTTPS-based notification receivers.

The following note shall be at the top of the registry:

This registry defines capabilities that can be supported by HTTPS-based notification receivers.

The fields for each registry are:

- o URN
 - * The name of the URN (required).
 - * The URN must conform to the syntax described by [RFC8141].
 - * The URN must begin with the string "urn:ietf:capability:https-notif-receiver".
- o Reference
 - * The RFC that defined the URN.
 - * The RFC must be in the form "RFC <Number>: <Title>".
- o Description
 - * An arbitrary description of the algorithm (optional).
 - * The description should be no more than a few sentences.
 - * The description is to be in English, but may contain UTF-8 characters as may be needed in some cases.

The update policy is either "RFC Required". Updates do not otherwise require an expert review by a Designated Expert.

Following is the initial assignment for this registry:

Record:

Name: urn:ietf:capability:https-notif-receiver:encoding:json
Reference: RFC XXXX
Description: Identifies support for JSON-encoded notifications.

Record:

Name: urn:ietf:capability:https-notif-receiver:encoding:xml
Reference: RFC XXXX
Description: Identifies support for XML-encoded notifications.

Record:

Name: urn:ietf:capability:https-notif-receiver:encoding:rfc8639-enabled
Reference: RFC XXXX
Description: Identifies support for RFC 8639 state machine.

9. References

9.1. 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.
- [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>>.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <<https://www.rfc-editor.org/info/rfc5277>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <<https://www.rfc-editor.org/info/rfc7407>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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>>.

9.2. Informative references

- [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>>.

[RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <<https://www.rfc-editor.org/info/rfc8141>>.

Appendix A. Configuration Examples

This non-normative section shows two examples for how the "ietf-https-notif-transport" module can be used to configure a publisher to send notifications to a receiver.

In both examples, the Publisher, acting as an HTTPS client, is configured to send notifications to a receiver at address 192.0.2.1, port 443, and configures the "path" leaf value to "/some/path", with server certificates, and the corresponding trust store that is used to authenticate a connection.

A.1. Using Subscribed Notifications (RFC 8639)

This example shows how an RFC 8639 [RFC8639] based publisher can be configured to send notifications to a receiver.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<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-subscribed-notif-rec\
eivers">
      <receiver-instance>
        <name>global-receiver-def</name>
        <https-receiver
          xmlns="urn:ietf:params:xml:ns:yang:ietf-https-notif-tran\
sport"
          xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-\
to-name">
          <tls>
            <tcp-client-parameters>
              <remote-address>receiver.example.com</remote-address>
              <remote-port>443</remote-port>
            </tcp-client-parameters>
            <tls-client-parameters>
              <server-authentication>
                <ca-certs>
                  <local-definition>
                    <certificate>
```

```

        <name>Server Cert Issuer #1</name>
        <cert-data>base64encodedvalue==</cert-data>
      </certificate>
    </local-definition>
  </ca-certs>
</server-authentication>
</tls-client-parameters>
<http-client-parameters>
  <client-identity>
    <basic>
      <user-id>my-name</user-id>
      <cleartext-password>my-password</cleartext-passwor\
d>
    </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:ph="urn:ietf:params:xml:ns:yang:ietf-https-no\
tif-transport">ph:https</transport>
  <stream-subtree-filter>some-subtree-filter</stream-subtree-fil\
ter>
  <stream>some-stream</stream>
  <receivers>
    <receiver>
      <name>subscription-specific-receiver-def</name>
      <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:\
ietf-subscribed-notif-receivers">global-receiver-def</receiver-insta\
nce-ref>
    </receiver>
  </receivers>
</subscription>
</subscriptions>
<truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">

```

```

<certificate-bags>
  <certificate-bag>
    <name>explicitly-trusted-server-ca-certs</name>
    <description>
      Trust anchors (i.e. CA certs) that are used to
      authenticate connections to receivers. Receivers
      are authenticated if their certificate has a chain
      of trust to one of these CA certificates.
      certificates.
    </description>
    <certificate>
      <name>ca.example.com</name>
      <cert-data>base64encodedvalue==</cert-data>
    </certificate>
    <certificate>
      <name>Fred Flintstone</name>
      <cert-data>base64encodedvalue==</cert-data>
    </certificate>
  </certificate-bag>
</certificate-bags>
</truststore>
</config>

```

A.2. Not Using Subscribed Notifications

In the case that it is desired to use HTTPS-based notifications outside of Subscribed Notifications, an application-specific module would need to define the configuration for sending the notification.

Following is an example module. Note that the module is "uses" the "https-receiver-grouping" grouping from the "ietf-https-notif-transport" module.

```

module example-custom-module {
  yang-version 1.1;
  namespace "http://example.com/example-custom-module";
  prefix "custom";

  import ietf-https-notif-transport {
    prefix "hnt";
    reference
      "RFC XXXX:
       An HTTPS-based Transport for Configured Subscriptions";
  }

  organization
    "Example, Inc.";
}

```



```
contact
  "Support at example.com";

description
  "Example of module not using Subscribed Notifications module.";

revision "2021-02-22" {
  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";
      description
        "A list of HTTPS notif receivers.";
      leaf name {
        type string;
        description
          "A unique name for the https notif receiver.";
      }
      uses hnt:https-receiver-grouping;
    }
  }
}
```

Following is what the corresponding configuration looks like:

```
<?xml version="1.0" encoding="UTF-8"?>
<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>receiver.example.com</remote-address>
            <remote-port>443</remote-port>
          </tcp-client-parameters>
        </tls>
      </https-receiver>
    </https-receivers>
  </example-module>
</config>
```

```

    </tcp-client-parameters>
    <tls-client-parameters>
      <server-authentication>
        <ca-certs>
          <local-definition>
            <certificate>
              <name>Server Cert Issuer #1</name>
              <cert-data>base64encodedvalue==</cert-data>
            </certificate>
          </local-definition>
        </ca-certs>
      </server-authentication>
    </tls-client-parameters>
    <http-client-parameters>
      <client-identity>
        <basic>
          <user-id>my-name</user-id>
          <cleartext-password>my-password</cleartext-password>
        </basic>
      </client-identity>
      <path>/some/path</path>
    </http-client-parameters>
  </tls>
</https-receiver>
</https-receivers>
</example-module>
<truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
  <certificate-bags>
    <certificate-bag>
      <name>explicitly-trusted-server-ca-certs</name>
      <description>
        Trust anchors (i.e. CA certs) that are used to
        authenticate connections to receivers.  Receivers
        are authenticated if their certificate has a chain
        of trust to one of these CA certificates.
      </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>

```

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Eric Voit, Henning Rogge, Martin Bjorklund, Reshad Rahman, and Rob Wilton.

Authors' Addresses

Mahesh Jethanandani
Kloud Services

Email: mjethanandani@gmail.com

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

A YANG Data Model for a Keystore
draft-ietf-netconf-keystore-21

Abstract

This document defines a YANG module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted or hidden. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

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

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

- * "2021-02-10" --> the publication date of this draft

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

- * Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Relation to other RFCs	4
1.2. Specification Language	6
1.3. Terminology	6
1.4. Adherence to the NMDA	6
2. The "ietf-keystore" Module	6
2.1. Data Model Overview	6
2.2. Example Usage	14
2.3. YANG Module	26
3. Support for Built-in Keys	34
4. Encrypting Keys in Configuration	37
5. Security Considerations	41
5.1. Security of Data at Rest	41
5.2. Unconstrained Private Key Usage	41
5.3. The "ietf-keystore" YANG Module	41
6. IANA Considerations	42
6.1. The "IETF XML" Registry	42
6.2. The "YANG Module Names" Registry	42
7. References	42
7.1. Normative References	42

7.2. Informative References	43
Appendix A. Change Log	45
A.1. 00 to 01	45
A.2. 01 to 02	45
A.3. 02 to 03	45
A.4. 03 to 04	46
A.5. 04 to 05	46
A.6. 05 to 06	46
A.7. 06 to 07	46
A.8. 07 to 08	47
A.9. 08 to 09	47
A.10. 09 to 10	47
A.11. 10 to 11	47
A.12. 11 to 12	48
A.13. 12 to 13	48
A.14. 13 to 14	48
A.15. 14 to 15	48
A.16. 15 to 16	48
A.17. 16 to 17	49
A.18. 17 to 18	49
A.19. 18 to 19	49
A.20. 19 to 20	49
A.21. 20 to 21	50
Acknowledgements	50
Author's Address	50

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 initialized 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 behavior for systems to implement. To simplify implementation, advanced key formats may be selectively implemented.

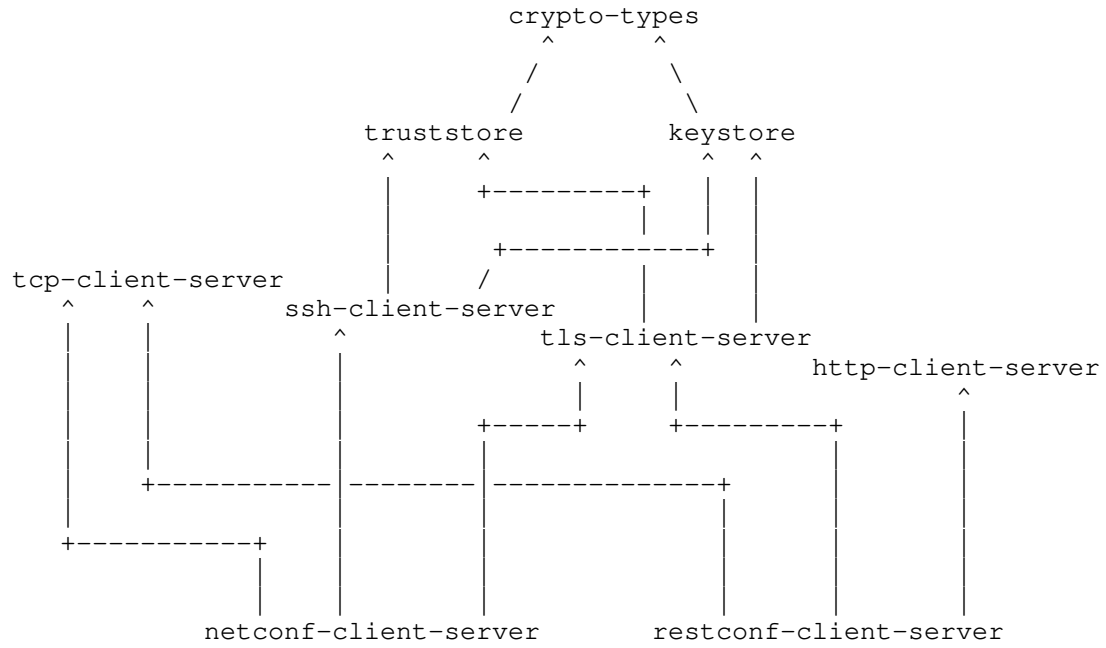
Implementations may utilize zero or more operating system level keystore utilities and/or hardware security modules (HSMs).

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, ultimately, enable the configuration of the 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 illustrating the module's use are provided in Section 2.2. The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-keystore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

2.1.1. Features

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

Features:

+-- 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, when the keystore module is implemented.
- * These typedefs are provided as an aid to downstream modules that import the "ietf-keystore" module.

2.1.3. Groupings

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

- * encrypted-by-choice-grouping
- * asymmetric-key-certificate-ref-grouping
- * local-or-keystore-symmetric-key-grouping
- * local-or-keystore-asymmetric-key-grouping
- * local-or-keystore-asymmetric-key-with-certs-grouping
- * local-or-keystore-end-entity-cert-with-key-grouping
- * keystore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "encrypted-by-choice-grouping" Grouping

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

```

| The grouping's name is intended to be parsed "(encrypted-
| by)-(choice)-(grouping)", not as "(encrypted)-(by-
| choice)-(grouping)".
|
grouping encrypted-by-choice-grouping
  +-- (encrypted-by-choice)
    +--:(symmetric-key-ref)
      | +-- symmetric-key-ref?    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.
- * This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations MUST augment in alternate "encrypted-by" references to the alternate locations.

2.1.3.2. The "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.
- * This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations MAY define an alternate grouping for references to the alternate locations.

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 solely 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 definition 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 solely 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 definition 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 solely 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 definition 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 solely 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 definition 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 structure for the symmetric and asymmetric keys is essentially the same, being a "list" inside a "container".
- * For asymmetric keys, each "asymmetric-key" uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 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

```

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-format          identityref
+--rw encrypted-value                      binary
+--rw certificates
+--rw certificate* [name]
+--rw name                                string
+--rw cert-data                          end-entity-cert-cms
+---n certificate-expiration
|   {certificate-expiration-notification}?
+-- expiration-date                      yang:date-and-time
+---x generate-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-format  identityref
+--rw encrypted-value         binary

```

Comments:

- * Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- * The protocol-accessible nodes for the "ietf-keystore" module are 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:one-symmetric-key-format</key-format>
    <encrypted-key>
      <encrypted-by>
        <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-k\
ey-ref>
      </encrypted-by>
      <encrypted-value-format>
        ct:cms-enveloped-data-format
      </encrypted-value-format>
      <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>
  </asymmetric-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: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-format>
          ct:cms-encrypted-data-format
        </encrypted-value-format>
        <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 usage'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-format identityref
|   |   |   |   |   |   |   |   +--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-format  identityref
  |   |   +--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-format      identityref
  |   |   |   |   |   +--rw encrypted-value            binary
  |   |   +--rw certificates
  |   |   |   +--rw certificate* [name]
  |   |   |   |   +--rw name                            string
  |   |   |   |   +--rw cert-data
  |   |   |   |   |   end-entity-cert-cms
  |   |   |   |   +--n certificate-expiration
  |   |   |   |   |   {certificate-expiration-notification}?
  |   |   |   |   |   +-- expiration-date              yang:date-and-time
  |   |   +--x generate-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-format      identityref
|   |   |   |   |   |   +---rw encrypted-value            binary
|   |   |   +---rw cert-data?
|   |   |   |   end-entity-cert-cms
|   |   +---n certificate-expiration
|   |   |   {certificate-expiration-notification}?
|   |   |   +--- expiration-date      yang:date-and-time
|   |   +---x generate-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 "2021-02-10" {
    description
      "Initial version";
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  container keystore-usage {
    description
      "An illustration of the various keystore groupings.";

    list symmetric-key {
      key name;
      leaf name {
```

```
        type string;
        description
            "An arbitrary name for this key.";
    }
    uses ks:local-or-keystore-symmetric-key-grouping;
    description
        "An symmetric key that may be configured locally or be a
        reference to a symmetric key in the keystore.";
}

list asymmetric-key {
    key name;
    leaf name {
        type string;
        description
            "An arbitrary name for this key.";
    }
    uses ks:local-or-keystore-asymmetric-key-grouping;
    description
        "An asymmetric key, with no certs, that may be configured
        locally or be a reference to an asymmetric key in the
        keystore. The intent is to reference just the asymmetric
        key, not any certificates that may also be associated
        with the asymmetric key.";
}

list asymmetric-key-with-certs {
    key name;
    leaf name {
        type string;
        description
            "An arbitrary name for this key.";
    }
    uses ks:local-or-keystore-asymmetric-key-with-certs-grouping;
    description
        "An asymmetric key and its associated certs, that may be
        configured locally or be a reference to an asymmetric key
        (and its associated certs) in the keystore.";
}

list end-entity-cert-with-key {
    key name;
    leaf name {
        type string;
        description
            "An arbitrary name for this key.";
    }
    uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
```

```
        description
          "An end-entity certificate and its associated 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@2021-02-10.yang"

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

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

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

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

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

  description
    "This module defines a 'keystore' to centralize management
     of security credentials.

    Copyright (c) 2020 IETF Trust and the persons identified
    as authors of the code. All rights reserved."
```

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC CCCC (<https://www.rfc-editor.org/info/rfcCCCC>); see the RFC itself for full legal notices.

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

```
revision 2021-02-10 {
  description
    "Initial version";
  reference
    "RFC CCCC: A YANG Data Model for a Keystore";
}

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

feature keystore-supported {
  description
    "The 'keystore-supported' feature indicates that the server
    supports the keystore.";
}

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

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

typedef symmetric-key-ref {
  type leafref {
    path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key"
```

```
        + "/ks:name";
    }
    description
        "This typedef enables modules to easily define a reference
        to a symmetric key stored in the keystore, when this
        module is implemented.";
}

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, when this
        module is implemented.";
}

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

grouping encrypted-by-choice-grouping {
    description
        "A grouping that defines a 'choice' statement that can be
        augmented into the 'encrypted-by' node, present in the
        'symmetric-key-grouping' and 'asymmetric-key-pair-grouping'
        groupings defined in RFC AAAA, enabling references to keys
        in the keystore, when this module is implemented.";
    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,
    when this module is implemented.";
  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.

    Servers that do not 'implement' this module, and hence
    'keystore-supported' is not defined, SHOULD augment in
    custom 'case' statements enabling references to the
    alternate keystore locations.";
  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, when this module is implmented.";
  }
}
}

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.

    Servers that do not 'implement' this module, and hence
    'keystore-supported' is not defined, SHOULD augment in
    custom 'case' statements enabling references to the
    alternate keystore locations.";
  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, when this module is implemented. 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.

    Servers that do not 'implement' this module, and hence
    'keystore-supported' is not defined, SHOULD augment in
    custom 'case' statements enabling references to the
    alternate keystore locations.";
  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, when
          this module is implemented.";
      }
    }
  }
}
```

```
    }
  }
}

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.

    Servers that do not 'implement' this module, and hence
    'keystore-supported' is not defined, SHOULD augment in
    custom 'case' statements enabling references to the
    alternate keystore locations.";
  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, when this
          module is implemented.";
      }
    }
  }
}

grouping keystore-grouping {
  description
    "Grouping definition enables use in other contexts.  If ever
    done, implementations MUST augment new 'case' statements
    into the various local-or-keystore 'choice' statements to
```

```
    supply leafrefs to the model-specific location(s).";
  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 their associated built-in certificates) to be referenced by configuration, the referenced keys and associated certificates MUST first be copied into <running>.

Built-in keys that are "hidden" MUST be copied into <running> using the same key values, so that the server can bind them to the built-in entries.

Built-in keys that are "encrypted" MAY be copied into other parts of the configuration so long as they are otherwise unmodified (e.g., the "encrypted-by" reference cannot be altered).

Built-in keys that are "cleartext" 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/were built-in.

The built-in keys and built-in associated certificates are immutable by configuration operations. With exception to additional/custom certificates associated to a built-in key, servers MUST ignore attempts to modify any aspect of built-in keys and/or built-in associated certificates.

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 configuration at runtime.

4.2. Configuring Encrypted Keys

Each time a new key is configured, it SHOULD be encrypted by a KEK.

In "ietf-crypto-types" [I-D.ietf-netconf-crypto-types], the format for encrypted values is described by identity statements derived from the "symmetrically-encrypted-value-format" and "symmetrically-encrypted-value-format" identity statements.

Implementations SHOULD provide an API that simultaneously generates and encrypts a key (symmetric or asymmetric) using a KEK. Thusly newly generated key cleartext values may never be 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 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.

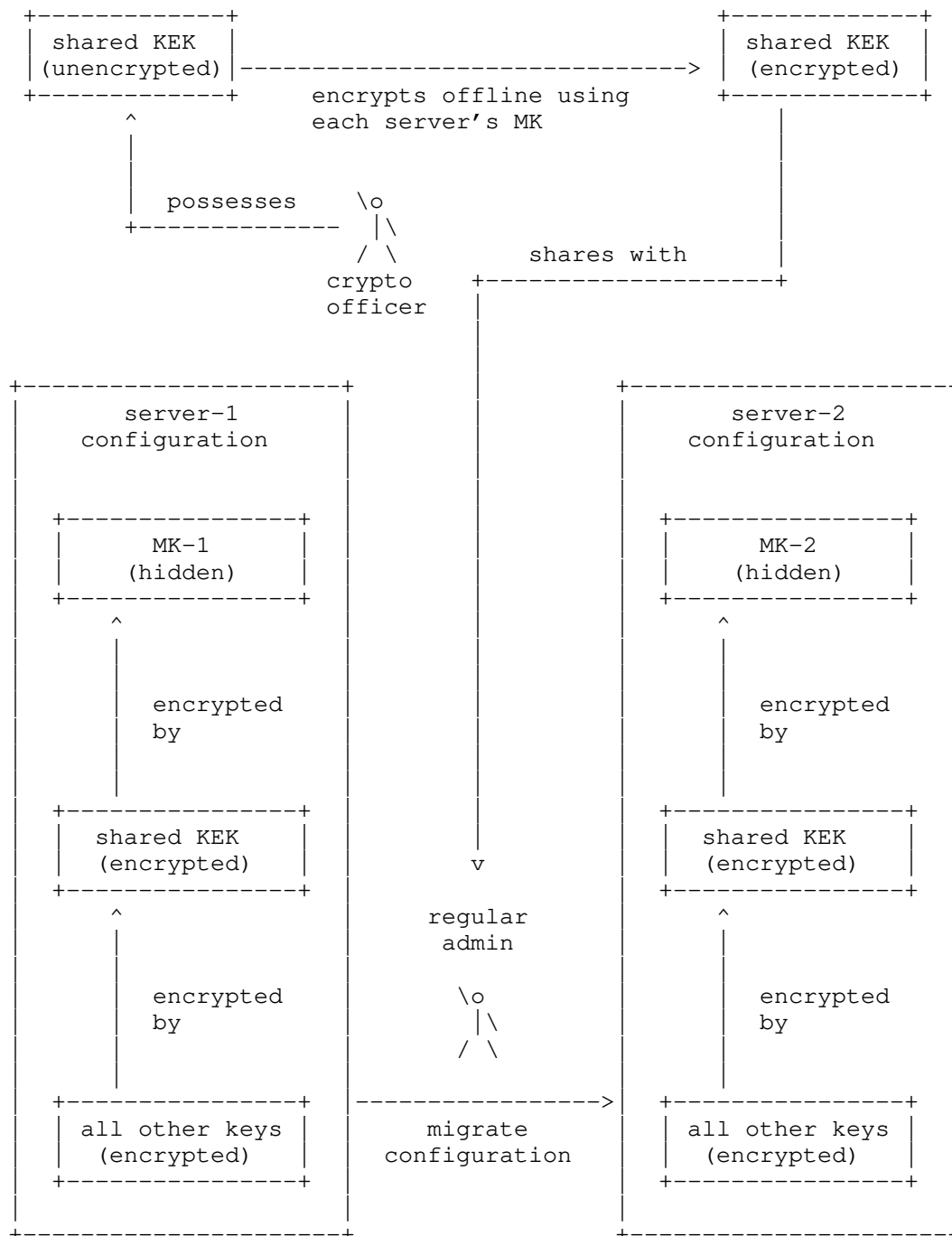
It is highly RECOMMENDED that MKs are built-in and hidden but, if this is not possible, highly restricted access mechanisms SHOULD be used to limit access to the MK's secret data to only highly authorized clients (e.g., an organization's crypto officer). In this case, it is RECOMMENDED that the MK is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



5. Security Considerations

5.1. Security of 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 5.3, 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. Unconstrained Private Key Usage

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

This module also does not constrain the usage of the associated public keys, other than in the context of a configured certificate (e.g., an identity certificate), in which case the key usage is constrained by the certificate.

5.3. 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 "rpc" or "action" statements, and thus the security considerations 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 IESG
XML: N/A, the requested URI is an XML namespace.

6.2. The "YANG Module Names" Registry

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

name: ietf-keystore
namespace: urn:ietf:params:xml:ns:yang:ietf-keystore
prefix: ks
reference: RFC CCCC

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]

Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-18, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>>.

[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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.

[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-netconf-client-server]

Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.

- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.
- [I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.
- [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-08, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.
- [I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.
- [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>>.
- [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 possibility 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 cryptotypes draft.

A.20. 19 to 20

- * Addressed SecDir comments from Magnus Nystroem and Sandra Murphy.

A.21. 20 to 21

- * Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir.
- * (Editorial) Removed the output of "grouping" statements in the tree diagrams for the "ietf-keystore" and "ex-keystore-usage" modules.
- * Addressed comments raised by YANG Doctor.

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: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-22

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:

* "2021-02-10" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	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	50
6.1.	Normative References	50
6.2.	Informative References	52
Appendix A.	Change Log	53
A.1.	00 to 01	53
A.2.	01 to 02	53
A.3.	02 to 03	54
A.4.	03 to 04	54
A.5.	04 to 05	54
A.6.	05 to 06	54
A.7.	06 to 07	54
A.8.	07 to 08	55
A.9.	08 to 09	55
A.10.	09 to 10	55
A.11.	10 to 11	55
A.12.	11 to 12	55
A.13.	12 to 13	56
A.14.	13 to 14	56
A.15.	14 to 15	56
A.16.	15 to 16	56
A.17.	16 to 17	56
A.18.	17 to 18	57
A.19.	18 to 19	57
A.20.	19 to 20	57
A.21.	20 to 21	57
A.22.	21 to 22	57
Acknowledgements	58
Author's Address	58

1. Introduction

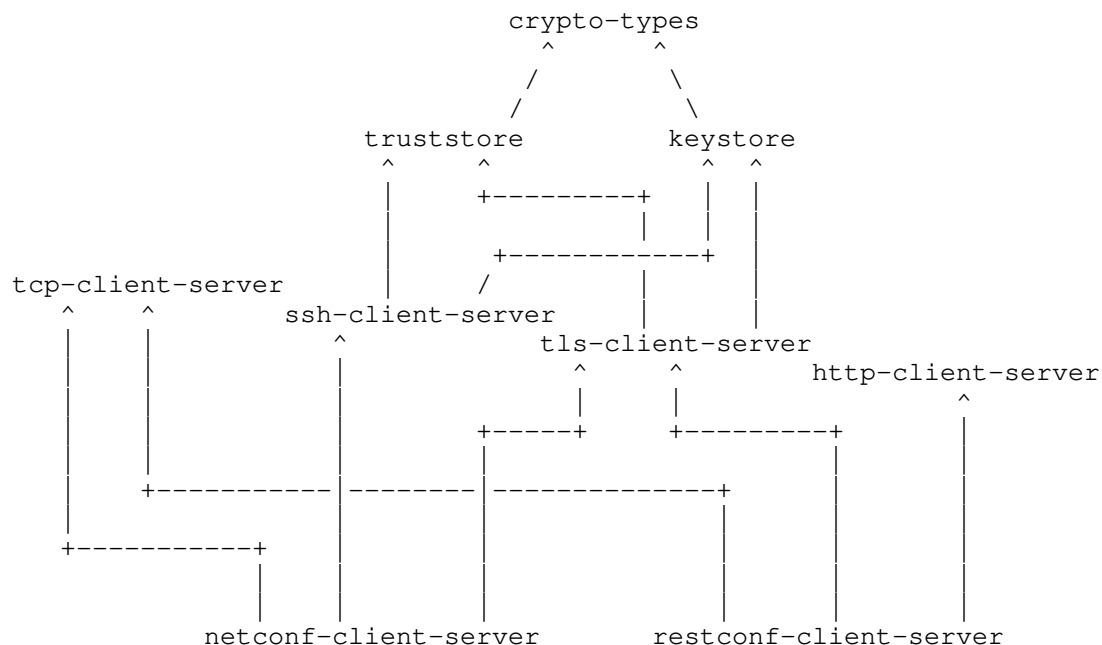
This document defines two YANG [RFC7950] modules, one module to configure a NETCONF [RFC6241] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [RFC6242] and NETCONF over TLS [RFC7589] and NETCONF Call Home connections [RFC8071].

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

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 "ietf-netconf-client" module defines the following "grouping" statements:

```
* netconf-client-grouping
* netconf-client-initiate-stack-grouping
* netconf-client-listen-stack-grouping
* netconf-client-app-grouping
```

Each of these groupings are presented in the following subsections.

2.1.2.1. The "netconf-client-grouping" Grouping

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

```
grouping netconf-client-grouping ---> <empty>
```

Comments:

```
* This grouping does not define any nodes, but is maintained so that
  downstream modules can augment nodes into it if needed.
```

- * The "netconf-client-grouping" defines, if it can be called that, the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

2.1.2.2. The "netconf-client-initiate-stack-grouping" Grouping

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

```

grouping netconf-client-initiate-stack-grouping
  +-- (transport)
    +--:(ssh) {ssh-initiate}?
      |   +-- ssh
      |     +-- tcp-client-parameters
      |       |   +---u tcpc:tcp-client-grouping
      |     +-- ssh-client-parameters
      |       |   +---u sshc:ssh-client-grouping
      |     +-- netconf-client-parameters
      |       |   +---u ncc:netconf-client-grouping
    +--:(tls) {tls-initiate}?
      |   +-- tls
      |     +-- tcp-client-parameters
      |       |   +---u tcpc:tcp-client-grouping
      |     +-- tls-client-parameters
      |       |   +---u tlsc:tls-client-grouping
      |     +-- netconf-client-parameters
      |       |   +---u ncc:netconf-client-grouping

```

Comments:

- * The "netconf-client-initiate-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that initiate connections to NETCONF servers, as opposed to receiving call-home [RFC8071] connections.
- * The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.
- * For the referenced grouping statement(s):
 - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
 - The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].

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

2.1.2.3. The "netconf-client-listen-stack-grouping" Grouping

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

```

grouping netconf-client-listen-stack-grouping
  +-- (transport)
    +--:(ssh) {ssh-listen}?
      +-- ssh
        +-- tcp-server-parameters
          | +---u tcps:tcp-server-grouping
        +-- ssh-client-parameters
          | +---u sshc:ssh-client-grouping
        +-- netconf-client-parameters
          +--u ncc:netconf-client-grouping
    +--:(tls) {tls-listen}?
      +-- tls
        +-- tcp-server-parameters
          | +---u tcps:tcp-server-grouping
        +-- tls-client-parameters
          | +---u tlsc:tls-client-grouping
        +-- netconf-client-parameters
          +---u ncc:netconf-client-grouping
  
```

Comments:

- * The "netconf-client-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that receive call-home [RFC8071] connections from NETCONF servers.
- * The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.
- * For the referenced grouping statement(s):
 - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
 - The "ssh-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-ssh-client-server].
 - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].

- The "netconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.4. The "netconf-client-app-grouping" Grouping

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

```

grouping netconf-client-app-grouping
+-- initiate! {ssh-initiate or tls-initiate}?
|   +-- netconf-server* [name]
|       +-- name? string
|       +-- endpoints
|           +-- endpoint* [name]
|               +-- name? string
|               +---u netconf-client-initiate-stack-grouping
|   +-- connection-type
|       +-- (connection-type)
|           +--:(persistent-connection)
|               | +-- persistent!
|           +--:(periodic-connection)
|               +-- periodic!
|                   +-- period? uint16
|                   +-- anchor-time? yang:date-and-time
|                   +-- idle-timeout? uint16
|       +-- reconnect-strategy
|           +-- start-with? enumeration
|           +-- max-attempts? uint8
+-- listen! {ssh-listen or tls-listen}?
    +-- idle-timeout? uint16
    +-- endpoint* [name]
        +-- name? string
        +---u netconf-client-listen-stack-grouping
  
```

Comments:

- * The "netconf-client-app-grouping" defines the configuration for a NETCONF client that supports both initiating connections to NETCONF servers as well as receiving call-home connections from NETCONF servers.
- * Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.
- * For the referenced grouping statement(s):
 - The "netconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.

- The "netconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

2.1.3. Protocol-accessible Nodes

The following 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</trustst\
ore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-server-ee-certs</trustst\
ore-reference>
          </ee-certs>
          <ssh-host-keys>
            <truststore-reference>trusted-ssh-public-keys</trustst\
ore-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</truststore-reference>
        </ca-certs>
        <ee-certs>
          <truststore-reference>trusted-server-ee-certs</truststore-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@2021-02-10.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 2021-02-10; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    revision-date 2021-02-10; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
    "WG Web:    <http://datatracker.ietf.org/wg/netconf/>
    WG List:    <mailto:netconf@ietf.org>
    Author:     Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:     Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module 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 2021-02-10 {
  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 than 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 "ietf-netconf-server" module defines the following "grouping" statements:

- * netconf-server-grouping
- * netconf-server-listen-stack-grouping
- * netconf-server-callhome-stack-grouping
- * netconf-server-app-grouping

Each of these groupings are presented in the following subsections.

3.1.2.1. The "netconf-server-grouping" Grouping

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

===== 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 initiating call-home connections to NETCONF clients.
- * Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.
- * For the referenced grouping statement(s):
 - The "netconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
 - The "netconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-netconf-server" module:

```

module: ietf-netconf-server
+--rw netconf-server
+---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</truststore-reference>
        </ca-certs>
        <ee-certs>
          <truststore-reference>trusted-client-ee-certs</truststore-reference>
        </ee-certs>
      </client-authentication>
      <keepalives>
        <peer-allowed-to-send/>
      </keepalives>

```

```

    </tls-server-parameters>
    <netconf-server-parameters>
      <client-identity-mappings>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>scooby-doo</name>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
      </client-identity-mappings>
    </netconf-server-parameters>
  </tls>
</endpoint>
</listen>

<!-- calling home to SSH and TLS based NETCONF clients -->
<call-home>
  <netconf-client> <!-- SSH-based client -->
    <name>config-mgr</name>
    <endpoints>
      <endpoint>
        <name>east-data-center</name>
        <ssh>
          <tcp-client-parameters>
            <remote-address>east.config-mgr.example.com</remote-ad\
dress>
            <keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </keepalives>
          </tcp-client-parameters>
          <ssh-server-parameters>
            <server-identity>
              <host-key>
                <name>deployment-specific-certificate</name>
                <public-key>
                  <keystore-reference>ssh-rsa-key</keystore-refere\
nce>
                </public-key>
              </host-key>
            </server-identity>
            <client-authentication>
              <supported-authentication-methods>

```

```

        <publickey/>
      </supported-authentication-methods>
    </client-authentication>
  </ssh-server-parameters>
  <netconf-server-parameters>
    <!-- nothing to configure -->
  </netconf-server-parameters>
</ssh>
</endpoint>
</endpoints>
<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>
  </ssh-server-parameters>
  <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>
  </tls>
</endpoint>

```



```

        </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@2021-02-10.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 2021-02-10; // stable grouping definitions
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

import ietf-ssh-server {
  prefix sshs;
  revision-date 2021-02-10; // stable grouping definitions
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

import ietf-tls-server {
  prefix tlss;
  revision-date 2021-02-10; // stable grouping definitions
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

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

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

Author: Kent Watsen <mailto:kent+ietf@watsen.net>
Author: Gary Wu <mailto:garywu@cisco.com>
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 2021-02-10 {

 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 IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

name: ietf-netconf-client
namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-client
prefix: ncc
reference: RFC HHHH

name: ietf-netconf-server
namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-server
prefix: ncs
reference: RFC HHHH

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore]

Watsen, K., "A YANG Data Model for a Keystore", 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-ssh-client-server]

Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.

[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-08, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.

[I-D.ietf-netconf-tls-client-server]

Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.

[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-18, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>>.
- [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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.
- [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-21, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-restconf-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-13, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-13>>.

[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-connection" description statements to be more like the RESTCONF draft, especially where it described dropping the underlying TCP connection.
- * Updated text to better reference where certain examples come from (e.g., which Section in which draft).
- * In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.
- * Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

A.13. 12 to 13

- * Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

A.14. 13 to 14

- * Adjusting from change in TLS client model (removing the top-level 'certificate' container), by swapping refining-in a 'mandatory true' statement with a 'must' statement outside the 'uses' statement.
- * Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

A.15. 14 to 15

- * Refactored both the client and server modules similar to how the ietf-restconf-server module was refactored in -13 of that draft, and the ietf-restconf-client grouping.

A.16. 15 to 16

- * Added refinement to make "cert-to-name/fingerprint" be mandatory false.
- * Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.

A.17. 16 to 17

- * Updated examples to include the "*-key-format" nodes.
- * Updated examples to remove the "required" nodes.
- * Updated examples to remove the "client-auth-defined-elsewhere" nodes.

A.18. 17 to 18

- * Updated examples to reflect new "bag" addition to truststore.

A.19. 18 to 19

- * Updated examples to remove the 'algorithm' nodes.
- * Updated examples to reflect the new TLS keepalives structure.
- * Added keepalives to the tcp-client-parameters section in the netconf-server SSH-based call-home example.
- * Added a TLS-based call-home example to the netconf-client example.
- * Added a "Note to Reviewers" note to first page.

A.20. 19 to 20

- * Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
- * Removed expanded tree diagrams that were listed in the Appendix.
- * Updated the Security Considerations section.

A.21. 20 to 21

- * Cleaned up titles in the IANA Considerations section
- * Fixed issues found by the SecDir review of the "keystore" draft.

A.22. 21 to 22

- * Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

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: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-22

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:

* "2021-02-10" --> 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 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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
---------------------------	---

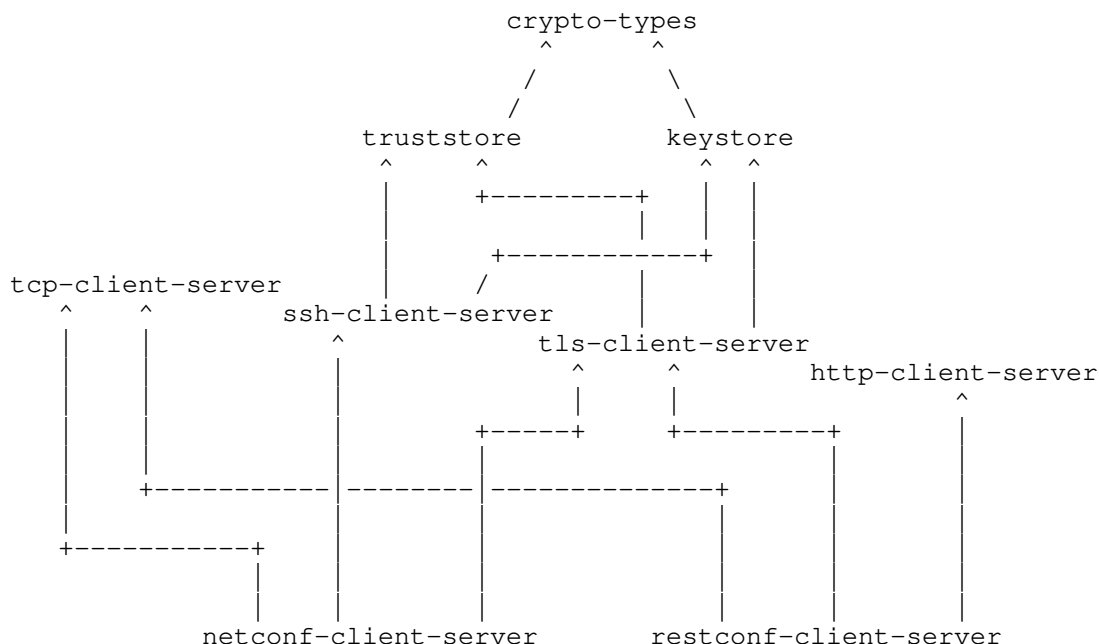
1.1.	Relation to other RFCs	4
1.2.	Specification Language	5
1.3.	Adherence to the NMDA	5
2.	The "ietf-restconf-client" Module	5
2.1.	Data Model Overview	6
2.2.	Example Usage	10
2.3.	YANG Module	14
3.	The "ietf-restconf-server" Module	24
3.1.	Data Model Overview	24
3.2.	Example Usage	29
3.3.	YANG Module	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
B.22.	21 to 22	55
Acknowledgements	55
Author's Address	55

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].

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the 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 is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

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 "ietf-restconf-client" module defines the following "grouping" statements:

```
* restconf-client-grouping
* restconf-client-initiate-stack-grouping
* restconf-client-listen-stack-grouping
* restconf-client-app-grouping
```

Each of these groupings are presented in the following subsections.

2.1.2.1. The "restconf-client-grouping" Grouping

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

```
grouping restconf-client-grouping ---> <empty>
```

Comments:

* This grouping does not define any nodes, but is maintained so that downstream modules can augment nodes into it if needed.

- * The "restconf-client-grouping" defines, if it can be called that, the configuration for just "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 2.1.2.2 and Section 2.1.2.3).

2.1.2.2. The "restconf-client-initiate-stack-grouping" Grouping

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

```
grouping restconf-client-initiate-stack-grouping
  +-- (transport)
    +--:(https) {https-initiate}?
      +-- https
        +-- tcp-client-parameters
          | +---u tcpc:tcp-client-grouping
        +-- tls-client-parameters
          | +---u tlsc:tls-client-grouping
        +-- http-client-parameters
          | +---u httpc:http-client-grouping
        +-- restconf-client-parameters
          +---u rcc:restconf-client-grouping
```

Comments:

- * The "restconf-client-initiate-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that initiate connections to RESTCONF servers, as opposed to receiving call-home [RFC8071] connections.
- * The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.
- * For the referenced grouping statement(s):
 - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
 - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
 - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].
 - The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.3. The "restconf-client-listen-stack-grouping" Grouping

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

```

grouping restconf-client-listen-stack-grouping
  +-- (transport)
    +--:(http) {http-listen}?
      +-- http
        +-- tcp-server-parameters
          | +---u tcps:tcp-server-grouping
        +-- http-client-parameters
          | +---u httpc:http-client-grouping
        +-- restconf-client-parameters
          +---u rcc:restconf-client-grouping
    +--:(https) {https-listen}?
      +-- https
        +-- tcp-server-parameters
          | +---u tcps:tcp-server-grouping
        +-- tls-client-parameters
          | +---u tlsc:tls-client-grouping
        +-- http-client-parameters
          | +---u httpc:http-client-grouping
        +-- restconf-client-parameters
          +---u rcc:restconf-client-grouping
  
```

Comments:

- * The "restconf-client-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that receive call-home [RFC8071] connections from RESTCONF servers.
- * The "transport" choice node enables both the HTTP and HTTPS transports to be configured, with each option enabled by a "feature" statement. Note that RESTCONF requires HTTPS, the HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-client.
- * For the referenced grouping statement(s):
 - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [I-D.ietf-netconf-tcp-client-server].
 - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-tls-client-server].
 - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].

- The "restconf-client-grouping" grouping is discussed in Section 2.1.2.1 in this document.

2.1.2.4. The "restconf-client-app-grouping" Grouping

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

```

grouping restconf-client-app-grouping
+-- initiate! {https-initiate}?
|   +-- restconf-server* [name]
|       +-- name? string
|       +-- endpoints
|           +-- endpoint* [name]
|               +-- name? string
|               +---u restconf-client-initiate-stack-grouping
|   +-- connection-type
|       +-- (connection-type)
|           +--:(persistent-connection)
|               | +-- persistent!
|               +--:(periodic-connection)
|                   +-- periodic!
|                       +-- period? uint16
|                       +-- anchor-time? yang:date-and-time
|                       +-- idle-timeout? uint16
|   +-- reconnect-strategy
|       +-- start-with? enumeration
|       +-- max-attempts? uint8
+-- listen! {http-listen or https-listen}?
    +-- idle-timeout? uint16
    +-- endpoint* [name]
        +-- name? string
        +---u restconf-client-listen-stack-grouping
  
```

Comments:

- * The "restconf-client-app-grouping" defines the configuration for a RESTCONF client that supports both initiating connections to RESTCONF servers as well as receiving call-home connections from RESTCONF servers.
- * Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.
- * For the referenced grouping statement(s):
 - The "restconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.

- The "restconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.3 in this document.

2.1.3. Protocol-accessible Nodes

The following 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</truststore-reference>
          </ca-certs>
          <ee-certs>
            <truststore-reference>trusted-server-ee-certs</truststore-reference>
          </ee-certs>
        </server-authentication>
        <keepalives>
          <peer-allowed-to-send/>
        </keepalives>
      </tls-client-parameters>
      <http-client-parameters>
        <client-identity>
          <basic>
            <user-id>bob</user-id>
            <cleartext-password>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@2021-02-10.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 2021-02-10 {  
  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 balancer) 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
            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 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 "ietf-restconf-server" module defines the following "grouping" statements:

```
* restconf-server-grouping
* restconf-server-listen-stack-grouping
* restconf-server-callhome-stack-grouping
* restconf-server-app-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "restconf-server-grouping" Grouping

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

```
grouping restconf-server-grouping
  +-- client-identity-mappings
    +---u x509c2n:cert-to-name
```

Comments:

- * The "restconf-server-grouping" defines the configuration for 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 initiating call-home connections to RESTCONF clients.
- * Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.
- * For the referenced grouping statement(s):
 - The "restconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
 - The "restconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-restconf-server" module:

```
module: ietf-restconf-server
  +--rw restconf-server
    +---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</truststore-reference>
        </ca-certs>
        <ee-certs>
          <truststore-reference>trusted-client-ee-certs</truststore-reference>
        </ee-certs>
      </client-authentication>
      <keepalives>
        <peer-allowed-to-send/>
      </keepalives>
    </tls-server-parameters>
    <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>
  </https>
</endpoint>

```

```
</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@2021-02-10.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 2021-02-10 {
  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 IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:      ietf-restconf-client
namespace: urn:ietf:params:xml:ns:yang:ietf-restconf-client
prefix:    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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.
- [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-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-08, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.
- [I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.

- [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-18, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>>.

- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.
- [I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.
- [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>>.
- [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.

XINSERT_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.

XINSERT_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-connection" 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 Considerations section
- * Fixed issues found by the SecDir review of the "keystore" draft.

B.22. 21 to 22

- * Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

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: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-23

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:

- * "2021-02-10" --> the publication date of this draft

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

- * Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. 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
A.23. 22 to 23	61
Acknowledgements	61
Author's Address	62

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.

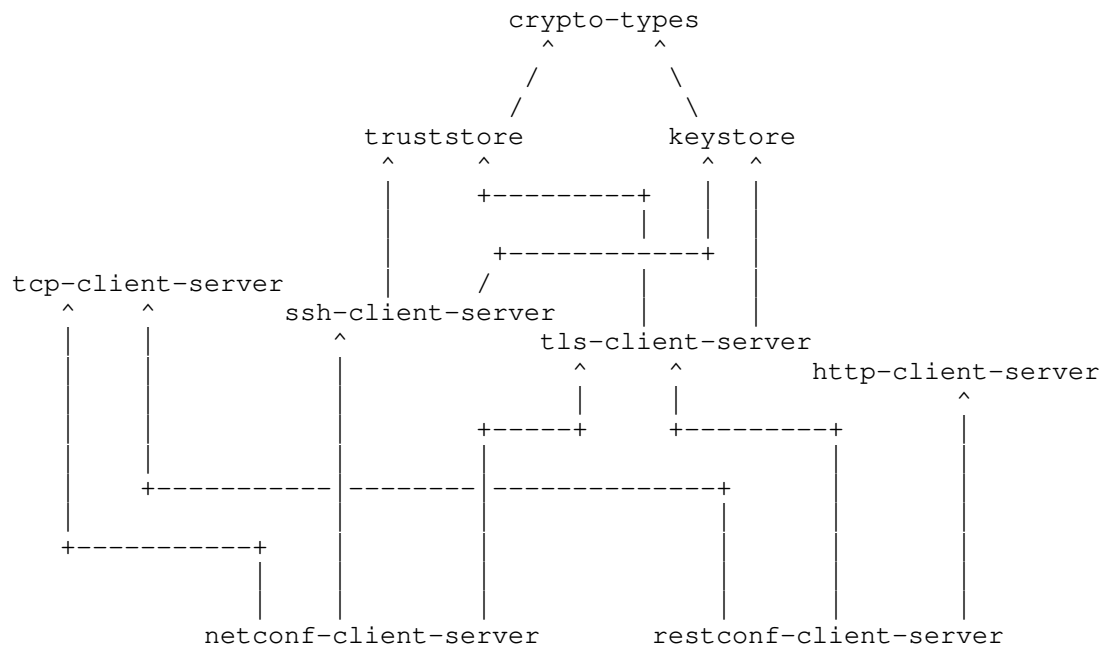
The modules defined in this document optionally support [RFC6187] enabling X.509v3 certificate based host keys and public keys.

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

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 oriented programming sense, in that they only define a "class" of algorithms, rather than a specific algorithm.

2.1.3. Groupings

The "ietf-ssh-common" module defines the following "grouping" statement:

- * transport-params-grouping

This grouping is presented in the following subsection.

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:alg="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
  <host-key>
    <host-key-alg>alg:x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>alg:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      alg:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>alg:aes256-ctr</encryption-alg>
    <encryption-alg>alg:aes192-ctr</encryption-alg>
    <encryption-alg>alg:aes128-ctr</encryption-alg>
    <encryption-alg>alg:aes256-cbc</encryption-alg>
    <encryption-alg>alg:aes192-cbc</encryption-alg>
    <encryption-alg>alg:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>alg:hmac-sha2-256</mac-alg>
    <mac-alg>alg: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@2021-02-10.yang"
```

```
module ietf-ssh-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix sshcmn;

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

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module defines a common features, identities, and
    groupings for Secure Shell (SSH).

    Copyright (c) 2020 IETF Trust and the persons identified
    as authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with
    or without modification, is permitted pursuant to, and
    subject to the license terms contained in, the Simplified
    BSD License set forth in Section 4.c of the IETF Trust's
    Legal Provisions Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC EEEE
    (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
    itself for full legal notices.;

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

  revision 2021-02-10 {
    description
      "Initial version";
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

  // Features
```



```
feature ssh-ecc {
  description
    "Elliptic Curve Cryptography is supported for SSH.";
  reference
    "RFC 5656: Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
}

feature ssh-x509-certs {
  description
    "X.509v3 certificates are supported for SSH per RFC 6187.";
  reference
    "RFC 6187: X.509v3 Certificates for Secure Shell
      Authentication";
}

feature ssh-dh-group-exchange {
  description
    "Diffie-Hellman Group Exchange is supported for SSH.";
  reference
    "RFC 4419: Diffie-Hellman Group Exchange for the
      Secure Shell (SSH) Transport Layer Protocol";
}

feature ssh-ctr {
  description
    "SDCTR encryption mode is supported for SSH.";
  reference
    "RFC 4344: The Secure Shell (SSH) Transport Layer
      Encryption Modes";
}

feature ssh-sha2 {
  description
    "The SHA2 family of cryptographic hash functions is
      supported for SSH.";
  reference
    "FIPS PUB 180-4: Secure Hash Standard (SHS)";
}

// Identities

identity public-key-alg-base {
  description
    "Base identity used to identify public key algorithms.";
}

identity ssh-dss {
```

```
    base public-key-alg-base;
    description
        "Digital Signature Algorithm using SHA-1 as the
        hashing algorithm.";
    reference
        "RFC 4253:
        The Secure Shell (SSH) Transport Layer Protocol";
}

identity ssh-rsa {
    base public-key-alg-base;
    description
        "RSASSA-PKCS1-v1_5 signature scheme using SHA-1 as the
        hashing algorithm.";
    reference
        "RFC 4253:
        The Secure Shell (SSH) Transport Layer Protocol";
}

identity ecdsa-sha2-nistp256 {
    if-feature "ssh-ecc and ssh-sha2";
    base public-key-alg-base;
    description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
        nistp256 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
}

identity ecdsa-sha2-nistp384 {
    if-feature "ssh-ecc and ssh-sha2";
    base public-key-alg-base;
    description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
        nistp384 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
}

identity ecdsa-sha2-nistp521 {
    if-feature "ssh-ecc and ssh-sha2";
    base public-key-alg-base;
    description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
        nistp521 curve and the SHA2 family of hashing algorithms.";
    reference
```

```
    "RFC 5656: Elliptic Curve Algorithm Integration in the
      Secure Shell Transport Layer";
  }

  identity x509v3-ssh-rsa {
    if-feature "ssh-x509-certs";
    base public-key-alg-base;
    description
      "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
       in an X.509v3 certificate and using SHA-1 as the hashing
       algorithm.";
    reference
      "RFC 6187: X.509v3 Certificates for Secure Shell
       Authentication";
  }

  identity x509v3-rsa2048-sha256 {
    if-feature "ssh-x509-certs and ssh-sha2";
    base public-key-alg-base;
    description
      "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
       in an X.509v3 certificate and using SHA-256 as the hashing
       algorithm. RSA keys conveyed using this format MUST have a
       modulus of at least 2048 bits.";
    reference
      "RFC 6187: X.509v3 Certificates for Secure Shell
       Authentication";
  }

  identity x509v3-ecdsa-sha2-nistp256 {
    if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
    base public-key-alg-base;
    description
      "Elliptic Curve Digital Signature Algorithm (ECDSA)
       using the nistp256 curve with a public key stored in
       an X.509v3 certificate and using the SHA2 family of
       hashing algorithms.";
    reference
      "RFC 6187: X.509v3 Certificates for Secure Shell
       Authentication";
  }

  identity x509v3-ecdsa-sha2-nistp384 {
    if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
    base public-key-alg-base;
    description
      "Elliptic Curve Digital Signature Algorithm (ECDSA)
       using the nistp384 curve with a public key stored in
```

```
        an X.509v3 certificate and using the SHA2 family of
        hashing algorithms.";
    reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
        Authentication";
}

identity x509v3-ecdsa-sha2-nistp521 {
    if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
    base public-key-alg-base;
    description
        "Elliptic Curve Digital Signature Algorithm (ECDSA)
        using the nistp521 curve with a public key stored in
        an X.509v3 certificate and using the SHA2 family of
        hashing algorithms.";
    reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
        Authentication";
}

identity key-exchange-alg-base {
    description
        "Base identity used to identify key exchange algorithms.";
}

identity diffie-hellman-group14-sha1 {
    base key-exchange-alg-base;
    description
        "Diffie-Hellman key exchange with SHA-1 as HASH and
        Oakley Group 14 (2048-bit MODP Group).";
    reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group-exchange-sha1 {
    if-feature "ssh-dh-group-exchange";
    base key-exchange-alg-base;
    description
        "Diffie-Hellman Group and Key Exchange with SHA-1 as HASH.";
    reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
        Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group-exchange-sha256 {
    if-feature "ssh-dh-group-exchange and ssh-sha2";
    base key-exchange-alg-base;
    description
```

```
        "Diffie-Hellman Group and Key Exchange with SHA-256 as HASH.";
    reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
            Secure Shell (SSH) Transport Layer Protocol";
}

identity ecdh-sha2-nistp256 {
    if-feature "ssh-ecc and ssh-sha2";
    base key-exchange-alg-base;
    description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
            nistp256 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
}

identity ecdh-sha2-nistp384 {
    if-feature "ssh-ecc and ssh-sha2";
    base key-exchange-alg-base;
    description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
            nistp384 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
}

identity ecdh-sha2-nistp521 {
    if-feature "ssh-ecc and ssh-sha2";
    base key-exchange-alg-base;
    description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
            nistp521 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
            Secure Shell Transport Layer";
}

identity encryption-alg-base {
    description
        "Base identity used to identify encryption algorithms.";
}

identity triple-des-cbc {
    base encryption-alg-base;
    description
        "Three-key 3DES in CBC mode.";
```

```
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity aes128-cbc {
    base encryption-alg-base;
    description
      "AES in CBC mode, with a 128-bit key.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity aes192-cbc {
    base encryption-alg-base;
    description
      "AES in CBC mode, with a 192-bit key.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity aes256-cbc {
    base encryption-alg-base;
    description
      "AES in CBC mode, with a 256-bit key.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity aes128-ctr {
    if-feature "ssh-ctr";
    base encryption-alg-base;
    description
      "AES in SDCTR mode, with 128-bit key.";
    reference
      "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
        Modes";
  }

  identity aes192-ctr {
    if-feature "ssh-ctr";
    base encryption-alg-base;
    description
      "AES in SDCTR mode, with 192-bit key.";
    reference
      "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
        Modes";
  }
}
```

```
identity aes256-ctr {
  if-feature "ssh-ctr";
  base encryption-alg-base;
  description
    "AES in SDCTR mode, with 256-bit key.";
  reference
    "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
      Modes";
}

identity mac-alg-base {
  description
    "Base identity used to identify message authentication
      code (MAC) algorithms.";
}

identity hmac-sha1 {
  base mac-alg-base;
  description
    "HMAC-SHA1";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

identity hmac-sha2-256 {
  if-feature "ssh-sha2";
  base mac-alg-base;
  description
    "HMAC-SHA2-256";
  reference
    "RFC 6668: SHA-2 Data Integrity Verification for the
      Secure Shell (SSH) Transport Layer Protocol";
}

identity hmac-sha2-512 {
  if-feature "ssh-sha2";
  base mac-alg-base;
  description
    "HMAC-SHA2-512";
  reference
    "RFC 6668: SHA-2 Data Integrity Verification for the
      Secure Shell (SSH) Transport Layer Protocol";
}

// Groupings

grouping transport-params-grouping {
  description
```

```

    "A reusable grouping for SSH transport parameters.";
reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
container host-key {
    description
        "Parameters regarding host key.";
    leaf-list host-key-alg {
        type identityref {
            base public-key-alg-base;
        }
        ordered-by user;
        description
            "Acceptable host key algorithms in order of descending
             preference.  The configured host key algorithms should
             be compatible with the algorithm used by the configured
             private key.  Please see Section 5 of RFC EEEE for
             valid combinations.

             If this leaf-list is not configured (has zero elements)
             the acceptable host key algorithms are implementation-
             defined.";
        reference
            "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
    }
}
container key-exchange {
    description
        "Parameters regarding key exchange.";
    leaf-list key-exchange-alg {
        type identityref {
            base key-exchange-alg-base;
        }
        ordered-by user;
        description
            "Acceptable key exchange algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable key exchange algorithms are implementation
             defined.";
    }
}
container encryption {
    description
        "Parameters regarding encryption.";
    leaf-list encryption-alg {
        type identityref {
            base encryption-alg-base;
        }
    }
}

```



```
    }
    ordered-by user;
    description
      "Acceptable encryption algorithms in order of descending
       preference.

       If this leaf-list is not configured (has zero elements)
       the acceptable encryption algorithms are implementation
       defined.";
  }
}
container mac {
  description
    "Parameters regarding message authentication code (MAC).";
  leaf-list mac-alg {
    type identityref {
      base mac-alg-base;
    }
  }
  ordered-by user;
  description
    "Acceptable MAC algorithms in order of descending
     preference.

     If this leaf-list is not configured (has zero elements)
     the acceptable MAC algorithms are implementation-
     defined.";
}
}
}
```

<CODE ENDS>

3. The "ietf-ssh-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-ssh-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-ssh-client" module in terms of its features and groupings.

3.1.1. Features

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

Features:

```
+-- ssh-client-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 "ietf-ssh-client" module defines the following "grouping" statement:

```
* ssh-client-grouping
```

This grouping is presented in the following subsection.

3.1.2.1. The "ssh-client-grouping" Grouping

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

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

```

grouping ssh-client-grouping
  +-- client-identity
  |   +-- username?      string
  |   +-- public-key! {client-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:alg="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 more 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@2021-02-10.yang"

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

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2021-02-10; // stable grouping definitions
    reference
      "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
  }

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

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
```


WG List: <mailto:netconf@ietf.org>
Author: Kent Watsen <mailto:kent+ietf@watsen.net>
Author: Gary Wu <mailto:garywu@cisco.com>;

description

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

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC EEEE (<https://www.rfc-editor.org/info/rfcEEEE>); see the RFC itself for full legal notices.;

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

```
revision 2021-02-10 {  
  description  
    "Initial version";  
  reference  
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";  
}
```

// Features

```
feature ssh-client-transport-params-config {  
  description  
    "SSH transport layer parameters are configurable on an SSH  
    client.";  
}
```

```
feature ssh-client-keepalives {  
  description  
    "Per socket SSH keepalive parameters are configurable for  
    SSH clients on the server implementing this feature.";
```

```

}

feature client-identity-password {
  description
    "Indicates that the 'password' authentication type
    is supported for client identification.";
}

feature client-identity-publickey {
  description
    "Indicates that the 'publickey' authentication type
    is supported for client identification.

    The 'publickey' authentication type is required by
    RFC 4252, but common implementations enable it to
    be disabled.";
}

feature client-identity-hostbased {
  description
    "Indicates that the 'hostbased' authentication type
    is supported for client identification.";
}

feature client-identity-none {
  description
    "Indicates that the 'none' authentication type is
    supported for client identification.";
}

// Groupings

grouping ssh-client-grouping {
  description
    "A reusable grouping for configuring a SSH client without
    any consideration for how an underlying TCP session is
    established.

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

  container client-identity {

```

```
nacm:default-deny-write;
must
  'public-key or password or hostbased or none or certificate';
description
  "The credentials that the client may use, pending
  the SSH server's requirements, by the SSH client
  to authenticate to the SSH server.";
leaf username {
  type string;
  description
    "The username of this user. This will be the username
    used, for instance, to log into an SSH server.";
}
container public-key {
  if-feature client-identity-publickey;
  presence
    "Indicates that publickey-based authentication
    is configured";
  description
    "A locally-defined or referenced asymmetric key
    pair to be used for client identification.";
  reference
    "RFC CCCC: A YANG Data Model for a Keystore";
  uses ks:local-or-keystore-asymmetric-key-grouping {
    refine "local-or-keystore/local/local-definition" {
      must 'public-key-format = "ct:ssh-public-key-format"';
    }
    refine "local-or-keystore/keystore/keystore-reference" {
      must 'deref(..)/ks:public-key-format'
        + ' = "ct:ssh-public-key-format"';
    }
  }
}
container password {
  if-feature client-identity-password;
  presence
    "Indicates that password-based authentication is
    configured.";
  description
    "A password to be used to authenticate 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 illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-ssh-server" module in terms of its features and groupings.

4.1.1. Features

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

Features:

```

+-- ssh-server-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 "ietf-ssh-server" module defines the following "grouping" statement:

* ssh-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "ssh-server-grouping" Grouping

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

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

```

grouping ssh-server-grouping
+-- server-identity
|   +-- host-key* [name]
|       +-- name?                string
|       +-- (host-key-type)
|           +--:(public-key)
|               +-- public-key
|                   +---u ks: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-key-format>
    </public-key>
</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:alg="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\
ruststore-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@2021-02-10.yang"
```

```

module ietf-ssh-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
  prefix sshs;

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

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

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

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

```

```
}

import ietf-keystore {
  prefix ks;
  reference
    "RFC CCCC: A YANG Data Model for a Keystore";
}

import ietf-ssh-common {
  prefix sshcmn;
  revision-date 2021-02-10; // stable grouping definitions
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

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

contact
  "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
  WG List:   <mailto:netconf@ietf.org>
  Author:    Kent Watsen <mailto:kent+ietf@watsen.net>
  Author:    Gary Wu <mailto:garywu@cisco.com>";

description
  "This module defines reusable groupings for SSH servers that
  can be used as a basis for specific SSH server instances.

  Copyright (c) 2020 IETF Trust and the persons identified
  as authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with
  or without modification, is permitted pursuant to, and
  subject to the license terms contained in, the Simplified
  BSD License set forth in Section 4.c of the IETF Trust's
  Legal Provisions Relating to IETF Documents
  (https://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC EEEE
  (https://www.rfc-editor.org/info/rfcEEEE); see the RFC
  itself for full legal notices.;

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

```
revision 2021-02-10 {
  description
    "Initial version";
  reference
    "RFC EEEE: YANG Groupings for SSH Clients and SSH Servers";
}

// Features

feature ssh-server-transport-params-config {
  description
    "SSH transport layer parameters are configurable on an SSH
    server.";
}

feature ssh-server-keepalives {
  description
    "Per socket SSH keepalive parameters are configurable for
    SSH servers on the server implementing this feature.";
}

feature client-auth-config-supported {
  description
    "Indicates that the configuration for how to authenticate
    clients can be configured herein, as opposed to in an
    application specific location. That is, to support the
    consuming data models that prefer to place client
    authentication with client definitions, rather than
    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 IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

6.2. The "YANG Module Names" Registry

This document registers 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-18, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>>.
- [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>>.
- [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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.
- [I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-

netconf-ssh-client-server-22, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.

[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-08, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.

[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.

A.23. 22 to 23

- * Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

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
Intended status: Standards Track
Expires: 14 August 2021

K. Watsen
Watsen Networks
M. Scharf
Hochschule Esslingen
10 February 2021

YANG Groupings for TCP Clients and TCP Servers
draft-ietf-netconf-tcp-client-server-09

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:

* "2021-02-10" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	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	22
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	30
A.1. 00 to 01	30
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	31
A.8. 07 to 08	31
A.9. 08 to 09	32
Authors' Addresses	32

1. Introduction

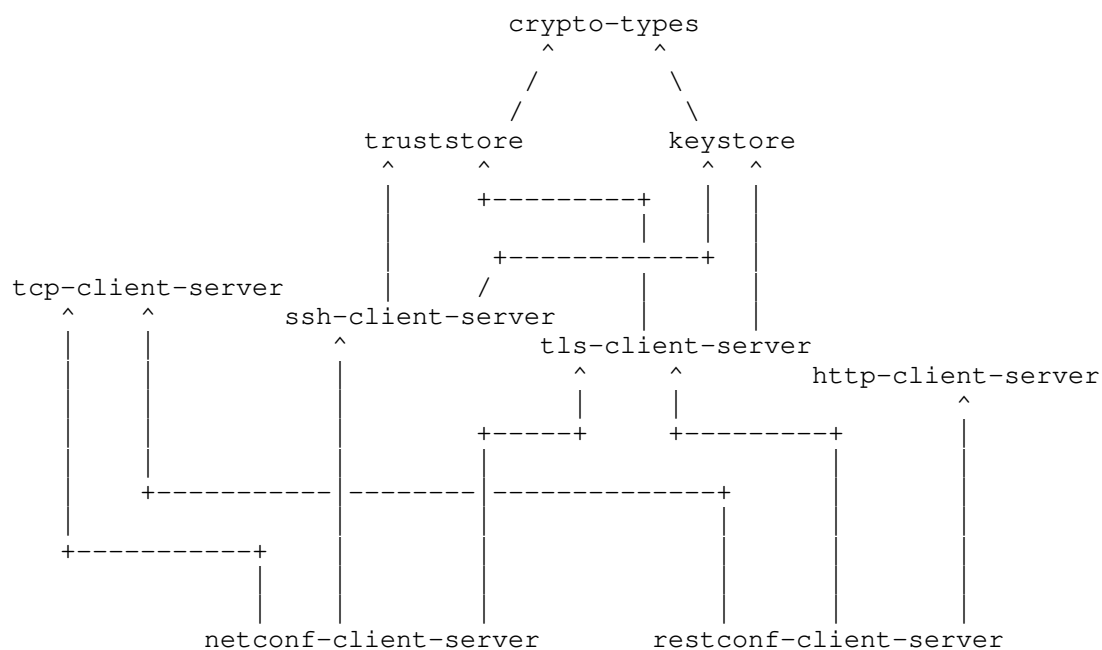
This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers, either as standalone or in conjunction with a stack protocol layer specific configurations.

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, ultimately, enable the configuration of the 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 is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

2. The "ietf-tcp-common" Module

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

2.1. Data Model Overview

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

2.1.1. Model Scope

This document 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 "ietf-tcp-common" module defines the following "grouping" statements:

- * tcp-common-grouping
- * tcp-connection-grouping

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.

2.1.1.5. 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@2021-02-10.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 2021-02-10 {
  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 descendant 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 illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-tcp-client" module in terms of its features and groupings.

3.1.1. Features

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

Features:

```
+-- local-binding-supported  
+-- tcp-client-keepalives  
+-- proxy-connect  
+-- 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 "ietf-tcp-client" module defines the following "grouping" statement:

```
* tcp-client-grouping
```

This grouping is presented in the following subsection.

3.1.2.1. The "tcp-client-grouping" Grouping

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

```

grouping tcp-client-grouping
+-- remote-address                inet:host
+-- remote-port?                  inet:port-number
+-- local-address?                inet:ip-address
|   {local-binding-supported}?
+-- local-port?                   inet:port-number
|   {local-binding-supported}?
+-- proxy-server! {proxy-connect}?
|   +-- (proxy-type)
|       +--:(socks4)
|           +-- socks4-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 prox-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@2021-02-10.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 2021-02-10 {
    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 illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-tcp-server" module in terms of its features and groupings.

4.1.1. Features

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

Features:

```
+-- tcp-server-keepalives
```

```

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

```

4.1.2. Groupings

The "ietf-tcp-server" module defines the following "grouping" statement:

```
* tcp-server-grouping
```

This grouping is presented in the following subsection.

4.1.2.1. The "tcp-server-grouping" Grouping

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

```

grouping tcp-server-grouping
  +-- local-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@2021-02-10.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 2021-02-10 {
  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 IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tcp-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tcp-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

6.2. The "YANG Module Names" Registry

This document registers 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-18, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-18>>.

[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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.

[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-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.

[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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.

[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-08, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.

[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>>.

[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. 07 to 08

- * Added missing IANA registration for "ietf-tcp-common"
- * Added "mandatory true" for the "username" and "password" leafs

- * Added an example of a TCP-client configured to connect via a proxy
- * Fixed issues found by the SecDir review of the "keystore" draft.
- * Updated the "ietf-tcp-client" module to use the new "password-grouping" grouping from the "crypto-types" module.

A.9. 08 to 09

- * Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

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: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-23

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:

- * "2021-02-10" --> the publication date of this draft

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

- * Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. 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	24
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
A.24. 22 to 23	58
Acknowledgements	58
Author's Address	59

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.

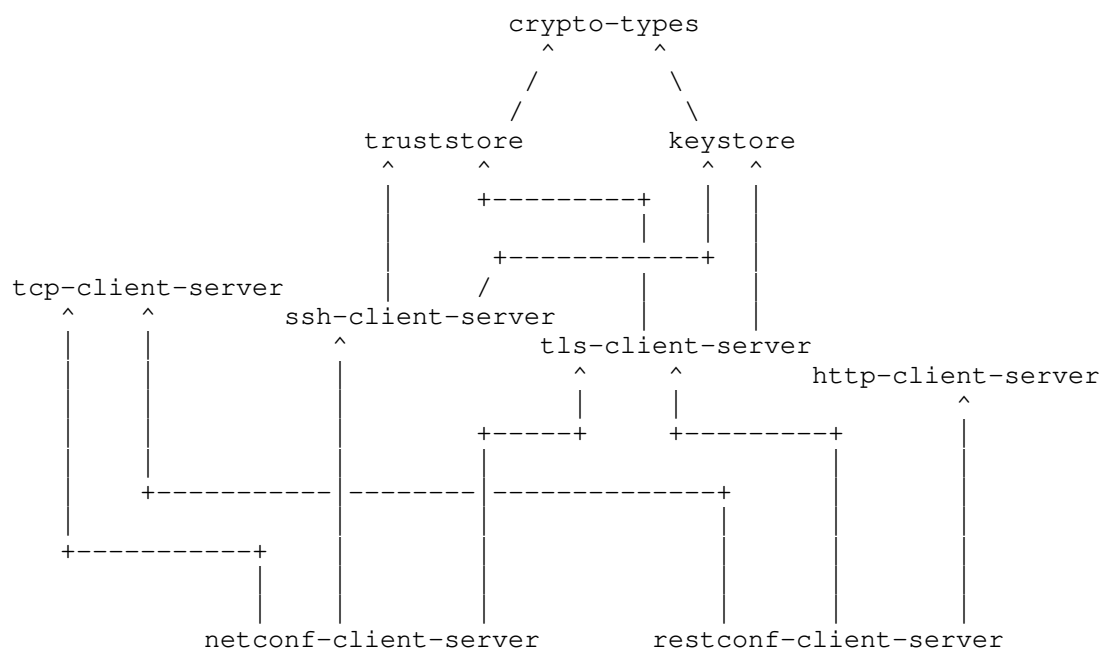
The client and server YANG modules in this document each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy 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, ultimately, enable the configuration of the 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 is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational>.

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, identities 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-edc-cbc-sha
    +-- ecdhe-rsa-with-3des-edc-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 "ietf-tls-common" module defines the following "grouping" statement:

```
* hello-params-grouping
```

This grouping is presented in the following subsection.

2.1.3.1. The "hello-params-grouping" Grouping

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

```
grouping hello-params-grouping
  +-- tls-versions
  |   +-- 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-edc-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@2021-02-10.yang"

```
module ietf-tls-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
  prefix tlscmn;

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

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module defines a common features, identities, and
    groupings for Transport Layer Security (TLS).

    Copyright (c) 2020 IETF Trust and the persons identified
```

as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC FFFF (<https://www.rfc-editor.org/info/rfcFFFF>); see the RFC itself for full legal notices.

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

```
revision 2021-02-10 {
  description
    "Initial version";
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

// Features

feature tls-1_0 {
  description
    "TLS Protocol Version 1.0 is supported.";
  reference
    "RFC 2246: The TLS Protocol Version 1.0";
}

feature tls-1_1 {
  description
    "TLS Protocol Version 1.1 is supported.";
  reference
    "RFC 4346: The Transport Layer Security (TLS) Protocol
      Version 1.1";
}

feature tls-1_2 {
  description
    "TLS Protocol Version 1.2 is supported.";
  reference
```

```
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
    }

    feature tls-1_3 {
        description
            "TLS Protocol Version 1.2 is supported.";
        reference
            "RFC 8446: The Transport Layer Security (TLS) Protocol
              Version 1.3";
    }

    feature tls-ecc {
        description
            "Elliptic Curve Cryptography (ECC) is supported for TLS.";
        reference
            "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
              for Transport Layer Security (TLS)";
    }

    feature tls-dhe {
        description
            "Ephemeral Diffie-Hellman key exchange is supported for TLS.";
        reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
              Version 1.2";
    }

    feature tls-3des {
        description
            "The Triple-DES block cipher is supported for TLS.";
        reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
              Version 1.2";
    }

    feature tls-gcm {
        description
            "The Galois/Counter Mode authenticated encryption mode is
              supported for TLS.";
        reference
            "RFC 5288: AES Galois Counter Mode (GCM) Cipher Suites for
              TLS";
    }

    feature tls-sha2 {
        description
            "The SHA2 family of cryptographic hash functions is supported
```



```
        for TLS.";
    reference
        "FIPS PUB 180-4: Secure Hash Standard (SHS)";
}

// Identities

identity tls-version-base {
    description
        "Base identity used to identify TLS protocol versions.";
}

identity tls-1.0 {
    if-feature "tls-1_0";
    base tls-version-base;
    description
        "TLS Protocol Version 1.0.";
    reference
        "RFC 2246: The TLS Protocol Version 1.0";
}

identity tls-1.1 {
    if-feature "tls-1_1";
    base tls-version-base;
    description
        "TLS Protocol Version 1.1.";
    reference
        "RFC 4346: The Transport Layer Security (TLS) Protocol
            Version 1.1";
}

identity tls-1.2 {
    if-feature "tls-1_2";
    base tls-version-base;
    description
        "TLS Protocol Version 1.2.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
            Version 1.2";
}

identity cipher-suite-base {
    description
        "Base identity used to identify TLS cipher suites.";
}

identity rsa-with-aes-128-cbc-sha {
    base cipher-suite-base;
```

```
    description
      "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
        Version 1.2";
  }

  identity rsa-with-aes-256-cbc-sha {
    base cipher-suite-base;
    description
      "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
        Version 1.2";
  }

  identity rsa-with-aes-128-cbc-sha256 {
    if-feature "tls-sha2";
    base cipher-suite-base;
    description
      "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA256.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
        Version 1.2";
  }

  identity rsa-with-aes-256-cbc-sha256 {
    if-feature "tls-sha2";
    base cipher-suite-base;
    description
      "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA256.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
        Version 1.2";
  }

  identity dhe-rsa-with-aes-128-cbc-sha {
    if-feature "tls-dhe";
    base cipher-suite-base;
    description
      "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA.";
    reference
      "RFC 5246: The Transport Layer Security (TLS) Protocol
        Version 1.2";
  }

  identity dhe-rsa-with-aes-256-cbc-sha {
    if-feature "tls-dhe";
```

```
base cipher-suite-base;
description
  "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA.";
reference
  "RFC 5246: The Transport Layer Security (TLS) Protocol
    Version 1.2";
}

identity dhe-rsa-with-aes-128-cbc-sha256 {
  if-feature "tls-dhe and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA256.";
  reference
    "RFC 5246: The Transport Layer Security (TLS) Protocol
      Version 1.2";
}

identity dhe-rsa-with-aes-256-cbc-sha256 {
  if-feature "tls-dhe and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA256.";
  reference
    "RFC 5246: The Transport Layer Security (TLS) Protocol
      Version 1.2";
}

identity ecdhe-ecdsa-with-aes-128-cbc-sha256 {
  if-feature "tls-ecc and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-ecdsa-with-aes-256-cbc-sha384 {
  if-feature "tls-ecc and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}
```

```
identity ecdhe-rsa-with-aes-128-cbc-sha256 {
  if-feature "tls-ecc and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-rsa-with-aes-256-cbc-sha384 {
  if-feature "tls-ecc and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-ecdsa-with-aes-128-gcm-sha256 {
  if-feature "tls-ecc and tls-gcm and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-ecdsa-with-aes-256-gcm-sha384 {
  if-feature "tls-ecc and tls-gcm and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-rsa-with-aes-128-gcm-sha256 {
  if-feature "tls-ecc and tls-gcm and tls-sha2";
  base cipher-suite-base;
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}
```

```

    }

    identity ecdhe-rsa-with-aes-256-gcm-sha384 {
        if-feature "tls-ecc and tls-gcm and tls-sha2";
        base cipher-suite-base;
        description
            "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.";
        reference
            "RFC 5289: TLS Elliptic Curve Cipher Suites with
              SHA-256/384 and AES Galois Counter Mode (GCM)";
    }

    identity rsa-with-3des-edc-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-edc-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 illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

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

3.1.1. Features

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

Features:

```
+-- tls-client-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 "ietf-tls-client" module defines the following "grouping" statement:

```
* tls-client-grouping
```

This grouping is presented in the following subsection.

3.1.2.1. The "tls-client-grouping" Grouping

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

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

```

grouping tls-client-grouping
+-- client-identity!
|   +-- (auth-type)
|       +--:(certificate) {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</trustst\
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@2021-02-10.yang"

module ietf-tls-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix tlsc;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2021-02-10; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
```

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

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC FFFF (<https://www.rfc-editor.org/info/rfcFFFF>); see the RFC itself for full legal notices.

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

```
revision 2021-02-10 {
  description
    "Initial version";
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

// Features

feature tls-client-hello-params-config {
  description
    "TLS hello message parameters are configurable on a TLS
    client.";
}

feature tls-client-keepalives {
  description
    "Per socket TLS keepalive parameters are configurable for
    TLS clients on the server implementing this feature.";
}

feature x509-certificate-auth {
  description
    "Indicates that the client supports authenticating servers
```

```

        using X.509 certificates.";
    }

    feature raw-public-key-auth {
        description
            "Indicates that the client supports authenticating servers
            using ray public keys.";
    }

    feature psk-auth {
        description
            "Indicates that the client supports authenticating servers
            using PSKs (pre-shared or pairwise-symmetric keys).";
    }

// Groupings

grouping tls-client-grouping {
    description
        "A reusable grouping for configuring a TLS client without
        any consideration for how an underlying TCP session is
        established.

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

    container client-identity {
        nacm:default-deny-write;
        presence
            "Indicates that TLS-level client authentication
            is sent. Present so that the 'choice' node's
            mandatory true doesn't imply that a client
            identity must be configured.";
        description
            "Identity credentials the TLS client MAY present when
            establishing a connection to a TLS server. If not
            configured, then client authentication is presumed to
            occur a protocol layer above TLS. When configured,
            and requested by the TLS server when establishing a
            TLS session, these credentials are passed in the

```

```

    Certificate message defined in Section 7.4.2 of
    RFC 5246.";
reference
  "RFC 5246: The Transport Layer Security (TLS) Protocol
  Version 1.2
  RFC CCCC: A YANG Data Model for a Keystore";
choice auth-type {
  mandatory true;
  description
    "A choice amongst available authentication types.";
  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 tlscomm: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 illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

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

4.1.1. Features

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

Features:

```
+-- tls-server-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 "ietf-tls-server" module defines the following "grouping" statement:

* tls-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "tls-server-grouping" Grouping

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

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

```

grouping tls-server-grouping
+-- server-identity
|   +-- (auth-type)
|       +--:(certificate) {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?
+-- test-peer-aliveness!
    +-- max-wait?
    +-- max-attempts?

```

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</trustst\
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@2021-02-10.yang"

```

module ietf-tls-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix tlss;

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

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

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

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2021-02-10; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>"

```

WG List: <mailto:netconf@ietf.org>
Author: Kent Watsen <mailto:kent+ietf@watsen.net>
Author: Gary Wu <mailto:garywu@cisco.com>;

description

"This module defines reusable groupings for TLS servers that can be used as a basis for specific TLS server instances.

Copyright (c) 2020 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC FFFF (<https://www.rfc-editor.org/info/rfcFFFF>); see the RFC itself for full legal notices.

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

```
revision 2021-02-10 {  
  description  
    "Initial version";  
  reference  
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";  
}
```

// Features

```
feature tls-server-hello-params-config {  
  description  
    "TLS hello message parameters are configurable on a TLS  
    server.";  
}
```

```
feature tls-server-keepalives {  
  description  
    "Per socket TLS keepalive parameters are configurable for  
    TLS servers on the server implementing this feature.";
```

```

}

feature client-auth-config-supported {
  description
    "Indicates that the configuration for how to authenticate
    clients can be configured herein, as opposed to in an
    application specific location. That is, to support the
    consuming data models that prefer to place client
    authentication with client definitions, rather than
    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 raw 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 IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

6.2. The "YANG Module Names" Registry

This document registers 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]

Watson, 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>>.

[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>>.

[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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.
- [I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.
- [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-08, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.

- [I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020,
<<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.
- [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 versions 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-authentication" 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.

A.24. 22 to 23

- * Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

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: 14 August 2021

K. Watsen
Watsen Networks
10 February 2021

A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-14

Abstract

This document defines a YANG module for configuring bags of certificates and bags of public keys that can be referenced by other data models for trust. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

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

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

- * "2021-02-10" --> the publication date of this draft

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

- * Appendix A. Change Log

Status of This Memo

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

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

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 14 August 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	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	6
2.2. Example Usage	12
2.3. YANG Module	21
3. Support for Built-in Trust Anchors	29
4. Security Considerations	32
4.1. Security of Data at Rest	32
4.2. Unconstrained Public Key Usage	32
4.3. The "ietf-truststore" YANG Module	32
5. IANA Considerations	33
5.1. The "IETF XML" Registry	33
5.2. The "YANG Module Names" Registry	33
6. References	33
6.1. Normative References	33
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	36
A.5.	04 to 05	36
A.6.	05 to 06	37
A.7.	06 to 07	37
A.8.	07 to 08	37
A.9.	08 to 09	37
A.10.	09 to 10	37
A.11.	10 to 11	38
A.12.	11 to 12	38
A.13.	12 to 13	38
A.14.	13 to 14	38
Acknowledgements		38
Author's Address		39

1. Introduction

This document defines a YANG 1.1 [RFC7950] module having the following characteristics:

Provide a central truststore for storing raw public keys and/or certificates.

Provide support for storing named bags of raw public keys and/or named bags of certificates.

Provide types that can be used to reference raw public keys or certificates stored in the central truststore.

Provide groupings that enable raw public keys and certificates to be configured locally or as references truststore instances.

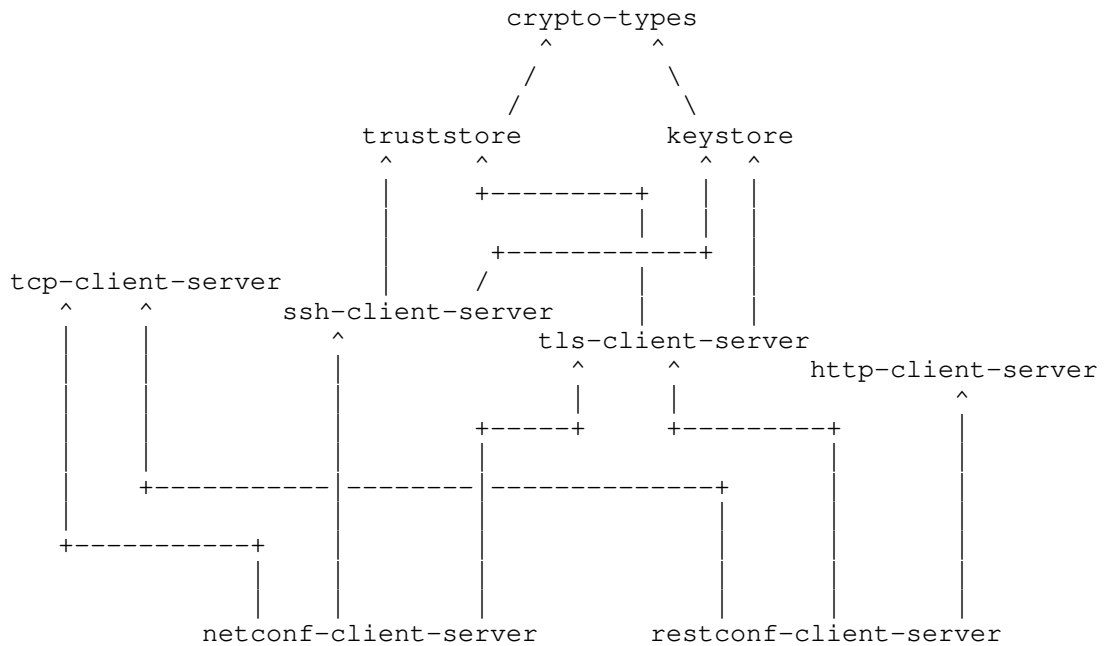
Enable the truststore to be instantiated in other data models, in addition to or in lieu of the central truststore instance.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, enable the configuration of the 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 is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, trust anchors installed during manufacturing (e.g., for trusted well-known services), are expected to appear in <operational> (see Section 3).

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 illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-truststore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

2.1.1. Features

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

Features:

```
+-- 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 in the truststore, when the truststore module is implemented.

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

2.1.3. Groupings

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

- * local-or-truststore-certs-grouping
- * local-or-truststore-public-keys-grouping
- * truststore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "local-or-truststore-certs-grouping" Grouping

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

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

Comments:

- * The "local-or-truststore-certs-grouping" grouping is provided solely 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 solely 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 defines a truststore instance as being composed of certificates and/or public keys, both of which are enabled by "feature" statements. The structure supporting certificates and public keys is essentially the same, having an outer list of "bags" containing in inner list of objects (certificates or public keys). The bags enable trust anchors serving a common purpose to be grouped and referenced together.
- * For certificates, each certificate is defined by the "trust-anchor-cert-grouping" grouping Section 2.1.4.7 of [I-D.ietf-netconf-crypto-types]. 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 protocol-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 the "truststore-grouping" exists separate from the protocol-accessible nodes definition is to enable instances of the truststore to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The examples in this section are encoded using XML, such as might be the case when using the NETCONF protocol. Other encodings MAY be used, such as JSON when using the RESTCONF protocol.

2.2.1. A Truststore Instance

This section presents an example illustrating trust anchors in <intended>, as per Section 2.1.4. Please see Section 3 for an example illustrating built-in values in <operational>.

The example contained in this section defines eight bags of trust anchors. There are four certificate-based bags and four public key based bags. The following diagram provides an overview of the contents in the example:

Certificate Bags

- +-- Trust anchor certs for authenticating a set of remote servers
- +-- End entity certs for authenticating a set of remote servers
- +-- Trust anchor certs for authenticating a set of remote clients
- +-- End entity certs for authenticating a set of remote clients

Public Key Bags

- +-- SSH keys to authenticate a set of remote SSH server
- +-- SSH keys to authenticate a set of remote SSH clients
- +-- Raw public keys to authenticate a set of remote SSH server
- +-- Raw public keys to authenticate a set of remote SSH clients

Following is the full example:

```
<truststore
  xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- A bag of Certificate Bags -->
  <certificate-bags>

    <!-- Trust Anchor Certs for Authenticating Servers -->
    <certificate-bag>
      <name>trusted-server-ca-certs</name>
      <description>
        Trust anchors (i.e. CA certs) used to authenticate server
        certificates.  A server certificate is authenticated if its
        end-entity certificate has a chain of trust to one of these
        certificates.
      </description>
      <certificate>
        <name>Server Cert Issuer #1</name>
        <cert-data>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>

<!-- Trust Anchor Certs for Authenticating Clients -->
<certificate-bag>
  <name>trusted-client-ca-certs</name>
  <description>
    Trust anchors (i.e. CA certs) used to authenticate client
    certificates. A client certificate is authenticated if its
    end-entity certificate has a chain of trust to one of these
    certificates.
  </description>
  <certificate>
    <name>Client Identity Issuer #1</name>
    <cert-data>base64encodedvalue==</cert-data>
  </certificate>
  <certificate>
    <name>Client Identity Issuer #2</name>
    <cert-data>base64encodedvalue==</cert-data>
  </certificate>
</certificate-bag>

<!-- End Entity Certs for Authenticating Clients -->
<certificate-bag>
  <name>trusted-client-ee-certs</name>
  <description>
    Specific end-entity certificates used to authenticate client
    certificates. A client certificate is authenticated if its
    end-entity certificate is an exact match to one of these
    certificates.
  </description>
  <certificate>
    <name>George Jetson</name>
    <cert-data>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 "2021-02-10" {
    description
      "Initial version";
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  container truststore-usage {
    description
```

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

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

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

2.3. YANG Module

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

<CODE BEGINS> file "ietf-truststore@2021-02-10.yang"

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

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

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

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

contact
  "WG Web   : <http://datatracker.ietf.org/wg/netconf/>
  WG List   : <mailto:netconf@ietf.org>
  Author    : Kent Watsen <kent+ietf@watsen.net>";

description
  "This module defines a 'truststore' to centralize management
  of trust anchors including certificates and public keys.

  Copyright (c) 2020 IETF Trust and the persons identified
  as authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with
  or without modification, is permitted pursuant to, and
  subject to the license terms contained in, the Simplified
  BSD License set forth in Section 4.c of the IETF Trust's
  Legal Provisions Relating to IETF Documents
  (https://trustee.ietf.org/license-info).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 2021-02-10 {
  description
    "Initial version";
  reference
    "RFC BBBB: A YANG Data Model for a Truststore";
}

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

```

/*****/

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

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

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

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

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

typedef certificate-bag-ref {
  type leafref {
    path "/ts:truststore/ts:certificate-bags/"
      + "ts:certificate-bag/ts:name";
  }
  description
    "This typedef defines a reference to a certificate bag
    in the truststore, when this module is implemented.";
}

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

```
        in a certificate bag in the truststore, when this module
        is implemented. 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 defines a reference to a public key bag
        in the truststore, when this module is implemented.";
}

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 defines a reference to a specific public key
        in a public key bag in the truststore, when this module is
        implemented. 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.

        Servers that do not 'implement' this module, and hence
        'truststore-supported' is not defined, SHOULD augment
        in custom 'case' statements enabling references to the
        alternate truststore locations.";
    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, when this module is implemented.";
  }
}
}
```

```
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.

     Servers that do not 'implement' this module, and hence
```



```
    'truststore-supported' is not defined, SHOULD augment
    in custom 'case' statements enabling references to the
    alternate truststore locations.";
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 exists
        in the truststore, when this module is implemented.";
    }
  }
}

grouping truststore-grouping {
  description
    "A grouping definition that enables use in other contexts.
    Where used, implementations MUST augment new 'case'
    statements into the various local-or-truststore 'choice'
    statements to supply leafrefs to the model-specific
    location(s).";
  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 server, 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 Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        manufacturer-signed objects, such as TLS server certificates,
        vouchers, etc.
      </description>
      <certificate>
        <name>Manufacturer Root CA Cert</name>
        <cert-data>base64encodedvalue==</cert-data>
      </certificate>
    </certificate-bag>

    <certificate-bag or:origin="or:system">
      <name>Built-In Public Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        certificates issued by public certificate authorities,
        such as the end-entity certificate for web servers.
      </description>
      <certificate>
        <name>Public Root CA Cert 1</name>
        <cert-data>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 bags of trust anchors and/or their trust anchors to be referenced by configuration, they MUST first be copied into <running>.

The built-in bags and/or their trust anchors MUST be copied into <running> using the same "key" values if it is desired for the server to maintain/update them (e.g., a software update may update a bag of trusted public CA certificates used for TLS-client connections).

Built-in bags and/or their trust anchors 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/were built-in.

The built-in bags and/or their trust anchors are immutable by configuration operations. Servers MUST ignore attempts to modify any aspect of built-in bags and/or their trust anchors from <running>.

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 Trust Anchor Certificates</name>
      <description>
        Certificates built into the device for authenticating
        certificates issued by public certificate authorities,
        such as the end-entity certificate for web servers.

        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. Security of Data at Rest

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

Security controls for the API (i.e., data in motion) are discussed in Section 4.3, but controls for the data at rest 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 protected from unauthorized modifications when at rest.

4.2. Unconstrained Public Key Usage

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

This module also enables the configuration of certificates, where each certificate may constrain the usage of the public key according to local policy.

4.3. 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 "rpc" or "action" statements, and thus the security considerations for such is not provided here.

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-truststore
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

5.2. The "YANG Module Names" Registry

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

name: ietf-truststore
namespace: urn:ietf:params:xml:ns:yang:ietf-truststore
prefix: ts
reference: RFC BBBB

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]

Watson, 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>>.

- [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-05, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-05>>.
- [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-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-21>>.
- [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-21, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-21>>.

[I-D.ietf-netconf-ssh-client-server]

Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-22>>.

[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-08, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-08>>.

[I-D.ietf-netconf-tls-client-server]

Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-22, 20 August 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-22>>.

[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>>.

[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.

A.14. 13 to 14

- * Added an "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir.
- * Addressed comments raised by YANG Doctor.

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, Nick Hancock, and Yoav Nir.

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.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	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.

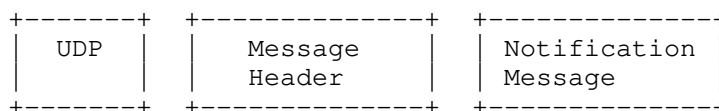


Figure 1: UDP-Notif Message Overview

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.

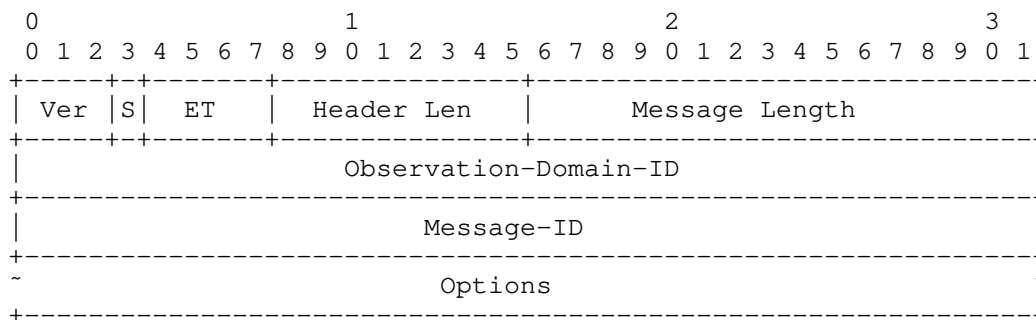


Figure 2: UDP-Notif Message Header Format

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.

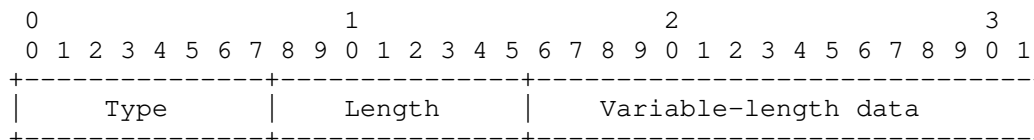


Figure 3: Generic Option Format

- * 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.

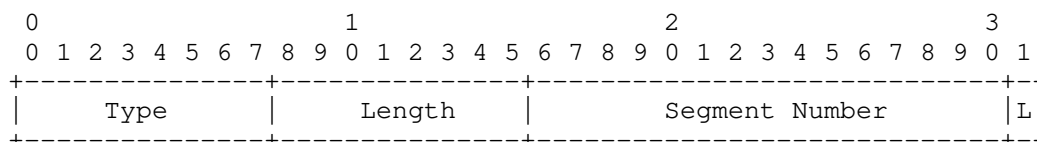


Figure 4: Segmentation Option Format

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 assigned.
- * **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 guidelines, 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 explicitly configure such a mode of operation.

Burst mitigation through packet pacing is RECOMMENDED. Disabling burst mitigation SHOULD require the users to explicitly 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: May 6, 2021

C. Feng
Q. Ma
Huawei
November 2, 2020

With System Capability for NETCONF
draft-ma-netconf-with-system-01

Abstract

The NETCONF protocol [RFC6241] defines ways to read configuration and state data from a NETCONF server. In some cases, a client-configured data item refers to a non-existent system generated data item (e.g., the auto-create interfaces ("eth1") is not yet present). In many situations, the system configured data item doesn't need to be known to the client and client-configured data item will automatically be removed from the operational state datastore and thus only appear in the intended datastore if client-configured data item doesn't exist. In other situations system configured data item needs to be known and overridden by the client. Not all server implementations treat the system configuration data in the same way. This document defines a capability-based extension to the NETCONF protocol that allows the NETCONF client to identify how system configuration are processed by the server, and also defines a new mechanism for client control of server processing of system configuration data.

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.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Terminology	3
1.2. Requirements Language	4
1.3. System Configuration Data Handling	4
2. System Configuration Datastore	4
2.1. Life Cycle of the system configuration	5
3. System Configuration data handling Basic Modes	5
3.1. 'report-all' Initialization During Reboot	6
3.2. 'report-all' <edit-config> Behavior	7
3.3. 'explicit' <edit-config> Behavior	7
4. Retrieval of System Configuration Data	8
5. With System Capability	8
5.1. Overview	8
5.2. Capability Identifier	9
5.3. Modifications to Existing Operations	9
5.3.1. <get> and <get-config> Operations	9
5.3.2. <edit-config> Operation	10
6. YANG Module for the <with-system> Parameter	10
7. IANA Considerations	13
8. Acknowledgements	14
9. References	14
9.1. Normative References	14
9.2. Informative References	14
Appendix A. Usage Examples	15
A.1. Example YANG Module	15
A.2. Example Data Set	15
A.3. Protocol Operation Examples	16
A.3.1. <with-system> = 'report-all'	17
A.3.2. <with-system> = 'report-all-tagged'	18
A.3.3. <with-system> = 'explicit'	20
A.3.4. <with-system> = 'trim'	21

Authors' Addresses	22
--------------------	----

1. Introduction

The NETCONF protocol [RFC6241] defines ways to read configuration and state data from a NETCONF server.

In some cases, a client-configured data item refers to a nonexistent system generated data item (e.g., the auto-create interfaces ("eth1") is not yet present).

- o In many situations, the system configured data item doesn't need to be known to the client and client-configured data item will automatically be removed from the operational state datastore and thus only appear in the intended datastore if client-configured data item doesn't exist.
- o In other situations system configured data item needs to be known and overridden by the client. Without system configuration datastore, the duplicated system configured data item in the system configuration need to be created and overridden by the client each time there is a system configured data item being referenced.

Therefore not all server implementations treat the system configuration data in the same way.

This document defines a capability-based extension to the NETCONF protocol that allows the NETCONF client to identify how system configuration are processed by the server, and also defines new mechanism for client control of server processing of system configuration data.

1.1. Terminology

This document assumes that the reader is familiar with the contents of [RFC6241], [RFC7950], [RFC8342], [RFC8407], and [RFC8525] and uses terminologies from those documents.

The following terms are defined in this document as follows:

System configuration: Configuration that is supplied by the device itself [RFC8342].

Logical resource dependent system configuration: When the device is powered on, the pre-provisioned configuration will be activated and provided, irrespective of physical resource present or not,

sometimes the pre-provisioned configuration will be provided unconditionally(e.g., loop back interface activation), sometimes not, e.g., only provided when a special functionality is enabled.

Physical resource dependent system configuration: When the device is powered on and the physical resource is present (e.g., insert interface card), the system will automatically detect it and load pre-provisioned configuration; when the physical resource is not present(remove interface card), the system configuration will be automatically cleared.

System configuration datastore: A configuration datastore holding the complete system configuration of the device. This datastore is referred to as "<system>".

1.2. Requirements Language

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

1.3. System Configuration Data Handling

The system configuration data handling behavior used by a server will impact NETCONF protocol operations in two ways:

- o Data retrieval: A server is normally allowed to exclude data nodes which it considers to contain the system configuration data. The actual nodes omitted depends on the system configuration data handling behavior used by the server.
- o Create and delete operations: The <edit-config> 'operation' attribute can be used to create and/or delete specific data nodes. These operations depend on whether the target node currently exists or not. The server's system configuration data handling behavior will determine whether the requested node currently exists in the configuration datastore or not.

2. System Configuration Datastore

Following guidelines for defining Datastores in the appendix A of [RFC8342], this document introduces a new datastore resource named 'system' that represents the pre-provisioned configuration or physical resource dependent configuration.

- o Name: "system"
- o YANG modules: all
- o YANG nodes: all "config true" data nodes
- o Management operations: The content of the datastore is set by the server in an implementation dependent manner. The content can not be changed by management operations via NETCONF, RESTCONF, the CLI etc unless specialized, dedicated operations are provided. The datastore can be read using the standard NETCONF/RESTCONF protocol operations.
- o Origin: This document does not define any new origin identity when it interacts with <operational> datastore. The system origin Metadata Annotation is used to indicate the origin of a data item.
- o Protocols: RESTCONF, NETCONF and other management protocol.
- o Defining YANG module: "ietf-netconf-with-system".

The datastore content is usually defined by the device vendor. It is static at most of time and MAY change e.g., depending on external factors like HW available or during device upgrade. <system> does not persist across reboots. It will be automatically loaded when the device is powered on or the physical resource is present.

2.1. Life Cycle of the system configuration

When the device is powered on, unconditional logical resource dependent system configuration will be generated and loaded into <system> automatically by the device operating system. Conditional logical resource dependent system configuration is only provided when a special functionality is enabled.

When the device is powered on and the physical resource is inserted into the device, physical resource dependent system configuration will be automatically loaded into <system>;

When the physical resource is removed from the device, the physical resource dependent system configuration will be automatically removed from <system>;

3. System Configuration data handling Basic Modes

Not all server implementations treat system configuration data in the same way. Instead of forcing a single implementation strategy, this

document allows a server to advertise a particular style of system configuration data handling, and the client can adjust accordingly.

NETCONF servers report system configuration data in different ways. This document specifies two standard defaults handling basic modes that a server implementor may choose from:

- o report-all
- o explicit

A server that uses the 'report-all' basic mode MUST automatically

- o Update <running> with the system configuration, after the "system" configuration has been altered as a consequence of a plug and play operation or device powering on operation. However the configurations in <running> can not be removed automatically when configuration data nodes in <system> is deleted since those configurations in <running> are likely to have already been modified or referenced.
- o The system configuration doesn't need to be explicitly set by the client first before the system configuration needs to be updated with client set configuration or referenced by client set configuration.

A server that uses the 'explicit' basic mode

MUST not update <running> with the system configuration,

The system configuration MUST be explicitly set by the client first before the system configuration needs to be updated with client set configuration or referenced by client set configuration.

3.1. 'report-all' Initialization During Reboot

At boot time, the device loads the saved system configuration into <running> together with saved startup configuration via 'merge' protocol operation. To save a new system configuration, data is copied to <system> via either implicit or explicit protocol operations.

The contents of <system> don't have to be persist across reboots. At each boot time, the device generates system configurations (e.g., unconditional logical resource dependent system configuration and physical resource dependent system configuration) and saves into <system>. Then the device loads the saved startup configuration into

<running>. The device may generate conditional logical resource dependent system configuration at the time of loading <startup>. Lastly, the device loads <system> into <running>. If there exists any conflict, the configuration in the <running> should succeed.

3.2. 'report-all' <edit-config> Behavior

The server MUST consider every data node to exist, even those set by the server.

- o A valid 'create' operation attribute for a data node that is loaded from <system> and explicitly set by the server MUST fail with a 'data-exists' error-tag;
- o A valid 'delete' operation attribute for a data node that is loaded from <system> and explicitly set by the server MUST succeed. The deleted system configuration MUST be reloaded into <running> immediately if the system configuration is still present in the <system>;
- o A valid 'merge' operation attribute for a data node that is loaded from <system> and explicitly set by the server MUST succeed.

3.3. 'explicit' <edit-config> Behavior

The server considers any data node that is explicitly set data to exist.

- o A valid 'create' operation attribute for a data node that is explicitly set by the server MUST succeed since the system configuration data is not present in the <running> configuration datastore.
- o A valid 'merge' operation attribute for a data node that is explicitly set by the server MUST succeed even though the name of data node in <system> is same as name of data node explicitly set by the client.
- o A valid 'delete' operation attribute for a data node that is explicitly set by the client MUST succeed even though the name of data node in <system> is same as name of data node explicitly set by the client. A valid 'delete' operation attribute for a data node that is not explicitly set by the client MUST fail since system configuration is not loaded into <running>.

4. Retrieval of System Configuration Data

When data is retrieved from a server using the 'report-all' basic mode, and the <with-system> parameter is not present, all data nodes MUST be reported including data nodes considered to be system configuration data by the server.

If the 'report-all' basic mode is used by the server and the <with-system> parameter supported by the server is set to a value equal to 'report-all', all data nodes MUST be reported, including any data nodes considered to be system configuration data by the server.

If the 'report-all' basic mode is used by the server and the <with-system> parameter supported by the server is set to a value equal to 'report-all-tagged', all data nodes MUST be reported, including any data nodes considered to be system configuration data by the server. Explicitly set data by the server will be tagged if the system configuration is applied.

When data is retrieved from a server using the 'explicit' basic mode, and the <with-system> parameter is not present, data nodes modified explicitly by the client MUST be reported.

If the 'explicit' basic mode is used by the server and the <with-system> parameter supported by the server is set to a value equal to 'explicit', data nodes MUST also be reported if explicitly modified by the client.

When data is retrieved from a server using the <with-system> parameter with a value equal to 'trim' , data nodes MUST be reported if considered to be not consistent with system configuration data by the server. Data node MUST NOT be reported if explicitly modified by the client.

5. With System Capability

5.1. Overview

The :with-system capability indicates which system-data-handling basic mode is supported by the server. These basic modes allow a NETCONF client to control whether system configuration data is returned by the server. Sending of system configuration data is controlled for each individual operation separately.

A NETCONF server implementing the :with-system capability:

- o MUST indicate its basic mode behavior by including the 'basic-mode' parameter in the capability URI;

- o MUST support the YANG module defined in Section 6 for the system configuration data handling mode indicated by the 'basic-mode' parameter.
- o SHOULD support the YANG module in Section 6 for the system configuration data handling mode identified by the 'report-all' or 'report-all-tagged' enumeration value.
- o If the 'report-all-tagged' system data handling mode is supported, then the 'origin' metadata attribute MUST be supported.
- o MAY support the YANG module in Section 6 for additional system data handling modes.

5.2. Capability Identifier

urn:ietf:params:netconf:capability:with-system:1.0

The identifier MUST have a parameter: "basic-mode". This indicates how the server will treat system configuration data, as defined in Section 3. The allowed values of this parameter are 'report-all', and 'explicit', as defined in Section 3.

The identifier MAY have another parameter: "also-supported". This parameter indicates which additional enumeration values (besides the basic-mode enumeration), the server will accept for the <with-system> parameter in Section 3. The value of the parameter is a comma separated list of one or more modes that are supported beside the mode indicated in the 'basic-mode' parameter. Possible modes are 'report-all', 'report-all-tagged', 'trim' and 'explicit', as defined in Section 3.

urn:ietf:params:netconf:capability:with-system:1.0?basic-mode=explicit&also-supported=report-all,report-all-tagged

5.3. Modifications to Existing Operations

5.3.1. <get> and <get-config> Operations

A new <with-system> XML element is added to the input for the <get>, <get-config> and <copy-config> operations. If the <with-system> element is present, it controls the reporting of system configuration data. The server MUST return system configuration data in the NETCONF <rpc-reply> messages according to the value of this element, if the server supports the specified retrieval mode (i.e., report-all/report-all-tagged).

This parameter only controls these specified retrieval operations, and does not impact any other operations or the non-volatile storage of configuration data.

The `<with-system>` element is defined in the XML namespace for the `ietf-netconf-with-system.yang` module in Section 6, not the XML namespace for the `<get>`, `<get-config>` and `<copy-config>` operations.

If the `<with-system>` element is not present, the server MUST follow its basic mode behavior as indicated by the `:with-system` capability identifier's `'basic-mode'` parameter, defined in Section 5.2.

The `<get>` and `<get-config>` operations support a separate filtering mechanism, using the `<filter>` parameter. The system configuration data filtering is conceptually done before the `<filter>` parameter is processed. For example, if the `<with-system>` parameter is equal to `'report-all'`, then the `<filter>` parameter is conceptually applied to all data nodes and all system configuration data.

5.3.2. `<edit-config>` Operation

The `<edit-config>` operation has several editing modes. The `'create'`, and `'delete'` editing operations are affected by the system configuration data handling basic mode. The other enumeration values for the NETCONF operation attribute are not affected.

If the operation attribute contains the value `'create'`, and the data node already exists in the target configuration datastore, then the server MUST return an `<rpc-error>` response with a `'invalid-value'` error-tag.

If the client sets a data node that is explicitly set by the server, the server MUST accept the request if it is valid. The server MUST keep or discard the new value based on its system configuration data handling basic mode.

6. YANG Module for the `<with-system>` Parameter

The following YANG module defines the addition of the `with-system` element to the `<get>`, `<get-config>`, and `<copy-config>` operations. The YANG language is defined in [RFC6020]. The above operations are defined in YANG in [RFC6241]. Every NETCONF server which supports the `:with-system` capability MUST implement this YANG module.

```
<CODE BEGINS> file="ietf-netconf-with-system@2019-12-31.yang"
module ietf-netconf-with-system {
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-with-system";
  prefix ncws;
```



```
import ietf-netconf { prefix nc; }

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

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

";

description
  "This module defines an extension to the NETCONF protocol
  that allows the NETCONF client to control how system configuration
  data are handled by the server in particular NETCONF operations.

  Copyright (c) 2010 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.";
// RFC Ed.: replace XXXX with actual RFC number and remove this note

revision 2019-12-31 {
  description
    "Initial version.";
  reference
    "RFC XXXX: With-system capability for NETCONF";
}

typedef with-system-mode {
  description
    "Possible modes to report system configuration data.";
  reference
    "RFC XXXX; section 3.";
    // RFC Ed.: replace XXXX with actual
    // RFC number and remove this note

  type enumeration {
    enum report-all {
      description
```

```
        "All system configuration data is reported.";
    reference
        "RFC XXXX; section 3.1";
        // RFC Ed.: replace XXXX with actual
        // RFC number and remove this note
    }
    enum report-all-tagged {
        description
            "All system configuration data is reported.
            Any nodes considered to be system configuration
            data will contain a 'origin' XML attribute,
            set to 'system'.";
        reference
            "RFC XXXX; section 3.4";
            // RFC Ed.: replace XXXX with actual
            // RFC number and remove this note
    }
    enum trim {
        description
            "Values are not reported if they contain the system
            configuration data.";
        reference
            "RFC XXXX; section 3.2";
            // RFC Ed.: replace XXXX with actual
            // RFC number and remove this note
    }
    enum explicit {
        description
            "Report values that contain the definition of
            explicitly set data.";
        reference
            "RFC XXXX; section 3.3";
            // RFC Ed.: replace XXXX with actual
            // RFC number and remove this note
    }
    }
}

grouping with-system-parameters {
    description
        "Contains the <with-system> parameter for control
        of system configuration data in NETCONF retrieval
        operations.";

    leaf with-system {
        description
```

```
        "The explicit system configuration data processing
        mode requested.";
    reference
        "RFC XXXX; section 4.6.1";
        // RFC Ed.: replace XXXX with actual
        // RFC number and remove this note

    type with-system-mode;
}

// extending the get-config operation
augment /nc:get-config/nc:input {
    description
        "Adds the <with-system> parameter to the
        input of the NETCONF <get-config> operation.";
    reference
        "RFC XXXX; section 4.6.1";
        // RFC Ed.: replace XXXX with actual
        // RFC number and remove this note

    uses with-system-parameters;
}

// extending the get operation
augment /nc:get/nc:input {
    description
        "Adds the <with-system> parameter to
        the input of the NETCONF <get> operation.";
    reference
        "RFC XXXX; section 4.6.1";
        // RFC Ed.: replace XXXX with actual
        // RFC number and remove this note

    uses with-system-parameters;
}
}
<CODE ENDS>
```

7. IANA Considerations

This document registers the following capability identifier URN in the 'Network Configuration Protocol Capability URNs registry':

urn:ietf:params:netconf:capability:with-system:1.0

This document registers two XML namespace URNs in the 'IETF XML registry', following the format defined in [RFC3688].

URI: urn:ietf:params:xml:ns:netconf:system:1.0
URI: urn:ietf:params:xml:ns:yang:ietf-netconf-with-system

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URIs are XML namespaces.

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC6020] .

name: ietf-netconf-with-system
prefix: ncws
namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-with-system
RFC: XXXX // RFC Ed.: replace XXXX and remove this comment

8. Acknowledgements

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

9.2. Informative References

- [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>>.
- [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>>.
- [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>>.
- [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <<https://www.rfc-editor.org/info/rfc8525>>.

Appendix A. Usage Examples

A.1. Example YANG Module

The following YANG module defines an example interfaces table to demonstrate how the <with-system> parameter behaves for a specific data model.

```
container interfaces {  
  list interface {  
    key name;  
    leaf name {  
      type string;  
    }  
    leaf description {  
      type string;  
    }  
    leaf-list ip-address {  
      type inet:ip-address;  
    }  
  }  
}
```

A.2. Example Data Set

The following data element shows the conceptual contents of the example server for the protocol operation examples in the next section. This includes all the configuration data nodes and system configuration leafs.

```

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="http://example.com/ns/interfaces">
    <interface>
      <name>lo0</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
    <interface>
      <name>lo1</name>
      <description>loopback</description>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::2</ip-address>
    </interface>
    <interface>
      <name>lo2</name>
      <description>loopback</description>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::3</ip-address>
    </interface>
    <interface>
      <name>lo3</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
  </interfaces>
</data>

```

In this example, the 'ip-address' field for each interface entry is set in the following manner:

name	ip-address	set by
lo0	127.0.0.1	server
lo0	::1	server
lo1	127.0.0.1	server
lo1	::2	client
lo2	127.0.0.1	server
lo2	::3	client
lo3	127.0.0.1	server
lo3	::1	server

A.3. Protocol Operation Examples

The following examples show some <get> operations using the 'with-system' element. The data model used for these examples is defined in Appendix A.1.

The client is retrieving all the data nodes within the 'interfaces' object, filtered with the <with-system> parameter.

A.3.1. <with-system> = 'report-all'

The behavior of the <with-system> parameter handling for the value 'report-all' is demonstrated in this example.

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <interfaces xmlns="http://example.com/ns/interfaces"/>
    </filter>
    <with-system
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
      report-all
    </with-system>
  </get>
</rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
  <interfaces xmlns="http://example.com/ns/interfaces">
    <interface>
      <name>lo0</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
    <interface>
      <name>lo1</name>
      <description>loopback</description>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::2</ip-address>
    </interface>
    <interface>
      <name>lo2</name>
      <description>loopback</description>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::3</ip-address>
    </interface>
    <interface>
      <name>lo3</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
  </interfaces>
</data>
</rpc-reply>
```

A.3.2. <with-system> = 'report-all-tagged'

The behavior of the <with-system> parameter handling for the value 'report-all-tagged' is demonstrated in this example. A 'tagged' data

node is an element that contains the 'origin' XML attribute, set to 'system'.

The actual data nodes tagged by the server depend on the system configuration data handling basic mode used by the server. Only the data nodes that are considered to be system configuration data will be tagged.

In this example, the server's basic mode is 'explicit', then only data nodes that are not explicitly set data are tagged. If the server's basic mode is 'report-all', then no data nodes are tagged.

```
<rpc message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <interfaces xmlns="http://example.com/ns/interfaces"/>
    </filter>
    <with-system
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
      report-all-tagged
    </with-system>
  </get>
</rpc>
```

```
<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
<data>
  <interfaces xmlns="http://example.com/ns/interfaces">
    <interface or:origin="or:system">
      <name>lo0</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
    <interface>
      <name>lo1</name>
      <description>loopback</description>
      <ip-address or:origin="or:system">127.0.0.1</ip-address>
      <ip-address>::2</ip-address>
    </interface>
    <interface>
      <name>lo2</name>
      <description>loopback</description>
      <ip-address or:origin="or:system">127.0.0.1</ip-address>
      <ip-address>::3</ip-address>
    </interface>
    <interface or:origin="or:system">
      <name>lo3</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
  </interfaces>
</data>
</rpc-reply>
```

A.3.3. <with-system> = 'explicit'

The behavior of the <with-system> parameter handling for the value 'explicit' is demonstrated in this example.

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <interfaces xmlns="http://example.com/ns/interfaces"/>
    </filter>
    <with-system
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
      explicit
    </with-system>
  </get>
</rpc>
```

```
<rpc-reply message-id="101"
            xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
  <interfaces xmlns="http://example.com/ns/interfaces">
    <interface>
      <name>lo0</name>
    </interface>
    <interface>
      <name>lo1</name>
      <description>loopback</description>
      <ip-address>::2</ip-address>
    </interface>
    <interface>
      <name>lo2</name>
      <description>loopback</description>
      <ip-address>::3</ip-address>
    </interface>
    <interface>
      <name>lo3</name>
    </interface>
  </interfaces>
</data>
</rpc-reply>
```

A.3.4. <with-system> = 'trim'

The behavior of the <with-system> parameter handling for the value 'trim' is demonstrated in this example.

```
<rpc message-id="104"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <interfaces xmlns="http://example.com/ns/interfaces"/>
    </filter>
    <with-system
      xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-system">
      trim
    </with-system>
  </get>
</rpc>

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
  <interfaces xmlns="http://example.com/ns/interfaces">
    <interface>
      <name>lo0</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
    <interface>
      <name>lo1</name>
      <description>loopback</description>
      <ip-address>127.0.0.1</ip-address>
    </interface>
    <interface>
      <name>lo2</name>
      <description>loopback</description>
      <ip-address>127.0.0.1</ip-address>
    </interface>
    <interface>
      <name>lo3</name>
      <ip-address>127.0.0.1</ip-address>
      <ip-address>::1</ip-address>
    </interface>
  </interfaces>
</data>
</rpc-reply>
```

Authors' Addresses

Feng Chong
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: frank.fengchong@huawei.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: maqiufang1@huawei.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 24, 2021

Q. Wu
Q. Ma
Huawei
P. Liu
China Mobile
January 20, 2021

Telemetry Data Export capability
draft-cao-netconf-data-export-capabilities-03

Abstract

This document proposes a YANG module for telemetry data export capabilities which augments system Capabilities model and provides additional telemetry data export attributes associated with system capabilities for transport dependent capability advertisement.

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 July 24, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Terminology	3
2. Data Export capability	3
2.1. Tree Diagram	4
3. YANG Module	4
4. IANA Considerations	10
4.1. Updates to the IETF XML Registry	10
4.2. Updates to the YANG Module Names Registry	10
5. Security Considerations	11
6. Contributors	11
7. References	12
7.1. Normative References	12
7.2. Informative References	13
Appendix A. Usage Example of interaction with Adaptive Subscription	14
Appendix B. Usage Example of interaction with UDP based Transport for Configured Subscription	16
Appendix C. Changes between Revisions	17
Authors' Addresses	18

1. Introduction

Notification capabilities 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 capabilities 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 the client and the server may still support various different transport specific parameters (e.g., transport protocol, encoding format, encryption). As described in section 3.1 of [RFC8641], a simple negotiation (i.e., inserting hints into error responses to a failed RPC request) between subscribers and publishers for subscription parameters increases the likelihood of success for subsequent RPC requests, but not guaranteed, which may cause unexpected failure or additional message exchange between client and server.

This document defines a corresponding solution that is built on top of [I-D.netconf-notification-capabilities]. Supplementing that work are YANG data model augmentations for transport dependent capability advertisement.

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:

- o A set of capabilities related to the amount of notifications the server can send out
- o Specification of which data nodes support on-change notifications.
- o 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.
- o 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, session level capabilities. They can be provided either at the implementation time or reported at the run time.

This document augments system Capabilities model and provides additional data export attributes associated with system capabilities:

- o Specification of transport protocol the client can use to establish a transport connection;
- o Specification of the encoding selection used (e.g., XML or JSON, Binary) for Data Modeled with YANG;
- o Specification of secure transport mechanisms that are needed by the client to communicate with the server;

- o Specification of the type of data compression algorithm (e.g., lossless data compression) the client can use for file compression and decompression
- o Specification of the notification message encapsulation type, either one notification per message or multiple notifications per message.
- o Specification of the update trigger type such as adaptive interval trigger, 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
  augment /sysc:system-capabilities/inc:subscription-capabilities:
    +--ro data-export-capabilities
      +--ro message-bundling-support?    boolean
  augment /sysc:system-capabilities/sysc:datastore-capabilities/sysc:per-node-capabilities:
    +--ro data-export-capabilities
      +--ro adaptive-interval-support    boolean
      +--ro timer-event-support?         boolean
      +--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;
  }
}

```

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 as transport protocol.";

}

identity udp-notif {

base transport-protocol;

description

"Identity for udp notif as transport protocol.";

reference

```
    "draft-ietf-netconf-udp-notif:UDP-based Transport
    for Configured Subscriptions";
}

identity http-notif {
    base transport-protocol;
    description
        "Identity for http notif as transport protocol.";
    reference
        "draft-ietf-netconf-https-notif: An HTTPS-based
        Transport for Configured Subscriptions";
}

identity grpc {
    base transport-protocol;
    description
        "Identity for grpc as transport protocol.";
}

identity security-protocol {
    description
        "Base identity for security protocol type.";
}

identity tls {
    base security-protocol;
    description
        "Identity for tls security protocol.";
}

identity ssh {
    base security-protocol;
    description
        "Identity for ssh transport protocol.";
}

identity encoding-format {
    description
        "Base identity for encoding format type.";
}

identity xml {
    base encoding-format;
    description
        "Identity for xml encoding format.";
}

identity json {
```

```
    base encoding-format;
    description
        "Identity for json encoding format.";
}

identity binary {
    base encoding-format;
    description
        "Identity for binary encoding format.";
}

identity gpb {
    base binary;
    description
        "Identity for gpb encoding format.";
}

identity cbor {
    base binary;
    description
        "Identity for cbor encoding format.";
}

identity compression-mode {
    description
        "Base identity for compression mode.";
}

identity gzip {
    base compression-mode;
    description
        "Identity for gzip compression mode.";
}

identity deflate {
    base compression-mode;
    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.";
    list data-export-capabilities {
        description
            "Capabilities related to telemetry data export capabilities 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.";
    }
  }
}
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.";
    }
  }
}
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 adaptive-interval-support {
      type boolean;
      default "false";
      description
        "Set to true if one event stream supports multiple period intervals and
        allows period interval switching. Set to false if the event stream doe
sn't
        support period interval switching.";
    }
    leaf timer-event-support {
      type boolean;
      default "false";
      description
        "Set to true if timer event is supported, i.e.,, schedule a specific ev
ent
        periodically with specified start time, duration, repeat option, repea
t
        interval.";
    }
    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 duplicated 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:

- o /sysc:system-capabilities/dec:transport-protocol
- o /sysc:system-capabilities/dec:encoding-format
- o /sysc:system-capabilities/dec:security-protocol
- o /sysc:system-capabilities/dec:compression-mode
- o /sysc:system-capabilities/inc:subscription-capabilities/
dec:message-bundling-support

6. Contributors

Ran Tao
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China
Email: taoran20@huawei.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

Thomas Graf
Swisscom
Binzring 17
Zuerich 8045
Switzerland

Email: thomas.graf@swisscom.com

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

7.2. 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-06 (work in progress), November 2020.
- [I-D.ietf-netconf-udp-notif]
Zheng, G., Zhou, T., Graf, T., Francois, P., and P. Lucente, "UDP-based Transport for Configured Subscriptions", draft-ietf-netconf-udp-notif-01 (work in progress), November 2020.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

Appendix A. Usage Example of interaction with 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 adaptive-interval-support and count-threshold-support are set to true.

```
<?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
        xmlns:dec="urn:ietf:params:xml:ns:yang:ietf-data-export-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:adaptive-interval-support>false</dec:adaptive-interval-support>
    <dec:threshold-event-support>false</dec:threshold-event-support>
  </dec:data-export-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:adaptive-interval-support>true</dec:adaptive-interval-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 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.

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <top xmlns="http://example.com/schema/1.2/config"
        xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push"
        >
        <yp:datastore
          xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
          ds:running
        </yp:datastore>
        <yp:datastore-xpath-filter
          xmlns: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:xpath-external-eval>in-errors &gt; 1000</as:xpath-external-eval>
            <as:watermark>1000</as:watermark>
            <as:period>5</as:period>
          </as:adaptive-period>
            <as:adaptive-period>
            <as:xpath-external-eval>in-errors &lt; 1000</as:xpath-external-eval>
            <as:watermark>1000</as:watermark>
            <as:period>60</as:period>
          </as:adaptive-period>
        </as:adaptive-subscriptions>
        </top>
      </config>
    </edit-config>
  </rpc>
```

Appendix B. Usage Example of interaction with UDP based Transport for Configured Subscription

```
<?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">
      <data-export-capabilities>
        <transport-protocol>udp</transport-protocol>
        <encoding-format>binary</encoding-format>
      </data-export-capabilities>
    </system-capabilities>
  </content-data>
</instance-data-set>
```

Appendix C. Changes between Revisions

v02 - v03

- o Change 'data-export-capabilities' into list type to support multiple transport protocol, encoding on the server.
- o Add Usage Example of interaction with UDP based Transport for Configured Subscription.
- o Add Thomas Graf as a contributor;
- o Update motivation in the introduction to clarify why this work is needed.
- o Support udp notif and http notif as two optional transport in the YANG data model.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: maqiufang1@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: liupengyjy@chinamobile.com