

Workgroup: NETCONF Working Group
Internet-Draft:
draft-ietf-netconf-netconf-client-server-21
Published: 20 August 2020
Intended Status: Standards Track
Expires: 21 February 2021
Authors: K. Watsen
Watsen Networks
NETCONF Client and Server Models

Abstract

This document defines two YANG modules, one module to configure a NETCONF client and the other module to configure a NETCONF server. Both modules support both the SSH and TLS transport protocols, and support both standard NETCONF and NETCONF Call Home connections.

Editorial Note (To be removed by RFC Editor)

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

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements (note: not all may be present):

- *AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
- *BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
- *CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
- *DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
- *EEEE --> the assigned RFC value for draft-ietf-netconf-ssh-client-server
- *FFFF --> the assigned RFC value for draft-ietf-netconf-tls-client-server
- *GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server
- *HHHH --> the assigned RFC value for this draft

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

*2020-08-20 --> the publication date of this draft

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

*[Appendix A](#). Change Log

Status of This Memo

This Internet-Draft 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 21 February 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](#)
 - 1.1. [Relation to other RFCs](#)
 - 1.2. [Specification Language](#)
 - 1.3. [Adherence to the NMDA](#)
2. [The "ietf-netconf-client" Module](#)
 - 2.1. [Data Model Overview](#)

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

1. Introduction

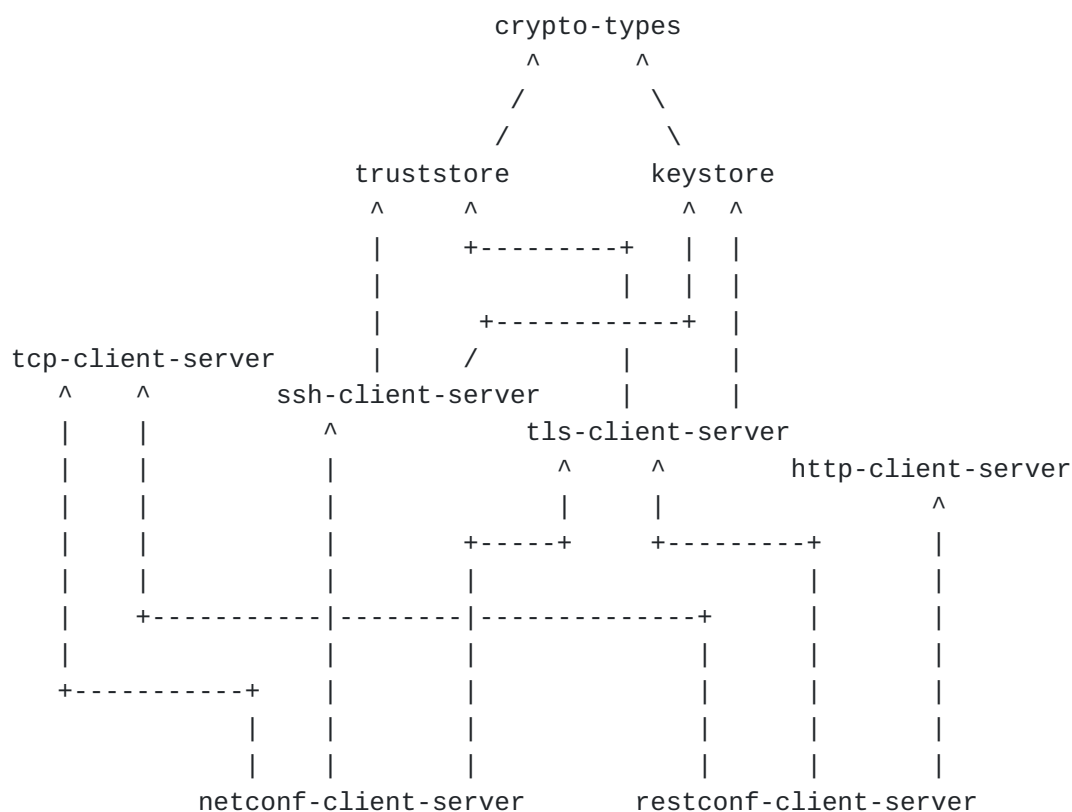
This document defines two YANG [[RFC7950](#)] modules, one module to configure a NETCONF [[RFC6241](#)] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [[RFC6242](#)] and NETCONF over TLS [[RFC7589](#)] and NETCONF Call Home connections [[RFC8071](#)].

1.1. Relation to other RFCs

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

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

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



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

http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

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

1.3. Adherence to the NMDA

This document 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 following diagram lists all the "grouping" statements defined in the "ietf-netconf-client" module:

Groupings:

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

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

Each of these groupings are presented in the following subsections.

2.1.2.1. The "netconf-client-grouping" Grouping

The following tree diagram [\[RFC8340\]](#) illustrates the "netconf-client-grouping" grouping:

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

Comments:

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

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

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

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

```

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

```

Comments:

*The "netconf-client-initiate-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that initiate connections to NETCONF servers, as opposed to receiving call-home [[RFC8071](#)] connections.

*The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

*For the referenced grouping statement(s):

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

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

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

```

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

```

Comments:

*The "netconf-client-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF clients that receive call-home [[RFC8071](#)] connections from NETCONF servers.

*The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

*For the referenced grouping statement(s):

- The "tcp-server-grouping" grouping is discussed in [Section 4.1.2.1](#) of [[I-D.ietf-netconf-tcp-client-server](#)].
- The "ssh-client-grouping" grouping is discussed in [Section 3.1.2.1](#) of [[I-D.ietf-netconf-ssh-client-server](#)].
- The "tls-client-grouping" grouping is discussed in [Section 3.1.2.1](#) of [[I-D.ietf-netconf-tls-client-server](#)].
- The "netconf-client-grouping" grouping is discussed in [Section 2.1.2.1](#) in this document.

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

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


```

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

```

Comments:

*The "netconf-client-app-grouping" defines the configuration for a NETCONF client that supports both initiating connections to NETCONF servers as well as receiving call-home connections from NETCONF servers.

*Both the "initiate" and "listen" subtrees must be enabled by "feature" statements.

*For the referenced grouping statement(s):

- The "netconf-client-initiate-stack-grouping" grouping is discussed in [Section 2.1.2.2](#) in this document.
- The "netconf-client-listen-stack-grouping" grouping is discussed in [Section 2.1.2.3](#) in this document.

2.1.3. Protocol-accessible Nodes

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

```
module: ietf-netconf-client
+--rw netconf-client
+---u netconf-client-app-grouping
```

Comments:

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

*For the "ietf-netconf-client" module, the protocol-accessible nodes are an instance of the "netconf-client-app-grouping" discussed in [Section 2.1.2.4](#) grouping.

*The reason for why "netconf-client-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of netconf-client-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

The following example illustrates configuring a NETCONF client to initiate connections, using both the SSH and TLS transport protocols, as well as to listen for call-home connections, again using both the SSH and TLS transport protocols.

This example is consistent with the examples presented in [Section 2.2](#) of [\[I-D.ietf-netconf-trust-anchors\]](#) and [Section 2.2](#) of [\[I-D.ietf-netconf-keystore\]](#).

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

```
<netconf-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-client"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- NETCONF servers to initiate connections to -->
  <initiate>
    <netconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <ssh>
            <tcp-client-parameters>
              <remote-address>corp-fw1.example.com</remote-address>
              <keepalives>
                <idle-time>15</idle-time>
                <max-probes>3</max-probes>
                <probe-interval>30</probe-interval>
              </keepalives>
            </tcp-client-parameters>
            <ssh-client-parameters>
              <client-identity>
                <username>foobar</username>
                <public-key>
                  <keystore-reference>ssh-rsa-key</keystore-referenc\
e>
                </public-key>
              </client-identity>
              <server-authentication>
                <ca-certs>
                  <truststore-reference>trusted-server-ca-certs</tru\
ststore-reference>
                </ca-certs>
                <ee-certs>
                  <truststore-reference>trusted-server-ee-certs</tru\
ststore-reference>
                </ee-certs>
              </server-authentication>
              <keepalives>
                <max-wait>30</max-wait>
                <max-attempts>3</max-attempts>
              </keepalives>
            </ssh-client-parameters>
            <netconf-client-parameters>
              <!-- nothing to configure -->
            </netconf-client-parameters>
          </ssh>
        </endpoint>
      </endpoints>
    </netconf-server>
  </initiate>
</netconf-client>
```

```

</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</truststore-reference>
                    </ca-certs>
                    <ee-certs>
                        <truststore-reference>trusted-server-ee-certs</truststore-reference>
                    </ee-certs>
                    <ssh-host-keys>
                        <truststore-reference>trusted-ssh-public-keys</truststore-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@2020-08-20.yang"
```

```

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

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

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

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

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

  import ietf-tls-client {
    prefix tlsc;
    revision-date 2020-08-20; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

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

  description
    "This module contains a collection of YANG definitions
    for configuring NETCONF clients."

```


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

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

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

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

```
revision 2020-08-20 {
  description
    "Initial version";
  reference
    "RFC HHHH: NETCONF Client and Server Models";
}

// Features

feature ssh-initiate {
  description
    "The 'ssh-initiate' feature indicates that the NETCONF client
    supports initiating SSH connections to NETCONF servers.";
  reference
    "RFC 6242:
    Using the NETCONF Protocol over Secure Shell (SSH)";
}

feature tls-initiate {
  description
    "The 'tls-initiate' feature indicates that the NETCONF client
    supports initiating TLS connections to NETCONF servers.";
  reference
    "RFC 7589: Using the NETCONF Protocol over Transport
    Layer Security (TLS) with Mutual X.509 Authentication";
}

feature ssh-listen {
  description
```

```

    "The 'ssh-listen' feature indicates that the NETCONF client
    supports opening a port to listen for incoming NETCONF
    server call-home SSH connections.";
reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

feature tls-listen {
    description
        "The 'tls-listen' feature indicates that the NETCONF client
        supports opening a port to listen for incoming NETCONF
        server call-home TLS connections.";
    reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

// Groupings

grouping netconf-client-grouping {
    description
        "A reusable grouping for configuring a NETCONF client
        without any consideration for how underlying transport
        sessions are established.

        This grouping currently doesn't define any nodes.";
}

grouping netconf-client-initiate-stack-grouping {
    description
        "A reusable grouping for configuring a NETCONF client
        'initiate' protocol stack for a single connection.";
    choice transport {
        mandatory true;
        description
            "Selects between available transports.";
        case ssh {
            if-feature "ssh-initiate";
            container ssh {
                description
                    "Specifies IP and SSH specific configuration
                    for the connection.";
                container tcp-client-parameters {
                    description
                        "A wrapper around the TCP client parameters
                        to avoid name collisions.";
                    uses tcpc:tcp-client-grouping {
                        refine "remote-port" {
                            default "830";
                            description

```

```

        "The NETCONF client will attempt to connect
        to the IANA-assigned well-known port value
        for 'netconf-ssh' (830) if no value is
        specified.";
    }
}
}
container ssh-client-parameters {
    description
        "A wrapper around the SSH client parameters to
        avoid name collisions.";
    uses sshc:ssh-client-grouping;
}
container netconf-client-parameters {
    description
        "A wrapper around the NETCONF client parameters
        to avoid name collisions.";
    uses ncc:netconf-client-grouping;
}
}
}
case tls {
    if-feature "tls-initiate";
    container tls {
        description
            "Specifies IP and TLS specific configuration
            for the connection.";
        container tcp-client-parameters {
            description
                "A wrapper around the TCP client parameters
                to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                    default "6513";
                    description
                        "The NETCONF client will attempt to connect
                        to the IANA-assigned well-known port value
                        for 'netconf-tls' (6513) if no value is
                        specified.";
                }
            }
        }
    }
}
container tls-client-parameters {
    must "client-identity" {
        description
            "NETCONF/TLS clients MUST pass some
            authentication credentials.";
    }
    description

```

```

        "A wrapper around the TLS client parameters
        to avoid name collisions.";
    uses tlsc:tls-client-grouping;
}
container netconf-client-parameters {
    description
        "A wrapper around the NETCONF client parameters
        to avoid name collisions.";
    uses ncc:netconf-client-grouping;
}
}
}
} // netconf-client-initiate-stack-grouping

grouping netconf-client-listen-stack-grouping {
    description
        "A reusable grouping for configuring a NETCONF client
        'listen' protocol stack for a single connection. The
        'listen' stack supports call home connections, as
        described in RFC 8071";
    reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    choice transport {
        mandatory true;
        description
            "Selects between available transports.";
        case ssh {
            if-feature "ssh-listen";
            container ssh {
                description
                    "SSH-specific listening configuration for inbound
                    connections.";
                container tcp-server-parameters {
                    description
                        "A wrapper around the TCP server parameters
                        to avoid name collisions.";
                    uses tcps:tcp-server-grouping {
                        refine "local-port" {
                            default "4334";
                            description
                                "The NETCONF client will listen on the IANA-
                                assigned well-known port for 'netconf-ch-ssh'
                                (4334) if no value is specified.";
                        }
                    }
                }
            }
        }
    }
    container ssh-client-parameters {
        description

```

```

        "A wrapper around the SSH client parameters
        to avoid name collisions.";
    uses sshc:ssh-client-grouping;
}
container netconf-client-parameters {
    description
        "A wrapper around the NETCONF client parameters
        to avoid name collisions.";
    uses ncc:netconf-client-grouping;
}
}
}
case tls {
    if-feature "tls-listen";
    container tls {
        description
            "TLS-specific listening configuration for inbound
            connections.";
        container tcp-server-parameters {
            description
                "A wrapper around the TCP server parameters
                to avoid name collisions.";
            uses tcps:tcp-server-grouping {
                refine "local-port" {
                    default "4334";
                    description
                        "The NETCONF client will listen on the IANA-
                        assigned well-known port for 'netconf-ch-ssh'
                        (4334) if no value is specified.";
                }
            }
        }
    }
    container tls-client-parameters {
        must "client-identity" {
            description
                "NETCONF/TLS clients MUST pass some
                authentication credentials.";
        }
        description
            "A wrapper around the TLS client parameters
            to avoid name collisions.";
        uses tlsc:tls-client-grouping;
    }
    container netconf-client-parameters {
        description
            "A wrapper around the NETCONF client parameters
            to avoid name collisions.";
        uses ncc:netconf-client-grouping;
    }
}

```

```

    }
  }
}
} // netconf-client-listen-stack-grouping

grouping netconf-client-app-grouping {
  description
    "A reusable grouping for configuring a NETCONF client
    application that supports both 'initiate' and 'listen'
    protocol stacks for a multiplicity of connections.";
  container initiate {
    if-feature "ssh-initiate or tls-initiate";
    presence "Enables client to initiate TCP connections";
    description
      "Configures client initiating underlying TCP connections.";
    list netconf-server {
      key "name";
      min-elements 1;
      description
        "List of NETCONF servers the NETCONF client is to
        maintain simultaneous connections with.";
      leaf name {
        type string;
        description
          "An arbitrary name for the NETCONF server.";
      }
    }
    container endpoints {
      description
        "Container for the list of endpoints.";
      list endpoint {
        key "name";
        min-elements 1;
        ordered-by user;
        description
          "A user-ordered list of endpoints that the NETCONF
          client will attempt to connect to in the specified
          sequence. Defining more than one enables
          high-availability.";
        leaf name {
          type string;
          description
            "An arbitrary name for the endpoint.";
        }
        uses netconf-client-initiate-stack-grouping;
      } // list endpoint
    } // container endpoints

    container connection-type {
      description

```

```

    "Indicates the NETCONF client's preference for how the
    NETCONF connection is maintained.";
choice connection-type {
  mandatory true;
  description
    "Selects between available connection types.";
  case persistent-connection {
    container persistent {
      presence "Indicates that a persistent connection is
        to be maintained.";
      description
        "Maintain a persistent connection to the NETCONF
        server. If the connection goes down, immediately
        start trying to reconnect to the NETCONF server,
        using the reconnection strategy.

        This connection type minimizes any NETCONF server
        to NETCONF client data-transfer delay, albeit at
        the expense of holding resources longer.";
    }
  }
  case periodic-connection {
    container periodic {
      presence "Indicates that a periodic connection is
        to be maintained.";
      description
        "Periodically connect to the NETCONF server.

        This connection type increases resource
        utilization, albeit with increased delay in
        NETCONF server to NETCONF client interactions.

        The NETCONF client should close the underlying
        TCP connection upon completing planned activities.

        In the case that the previous connection is still
        active, establishing a new connection is NOT
        RECOMMENDED.";
      leaf period {
        type uint16;
        units "minutes";
        default "60";
        description
          "Duration of time between periodic connections.";
      }
      leaf anchor-time {
        type yang:date-and-time {
          // constrained to minute-level granularity
          pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
        }
      }
    }
  }
}

```

```

        + '(Z|[\+\-]\d{2}:\d{2})';
    }
    description
        "Designates a timestamp before or after which a
        series of periodic connections are determined.
        The periodic connections occur at a whole
        multiple interval from the anchor time. For
        example, for an anchor time is 15 minutes past
        midnight and a period interval of 24 hours, then
        a periodic connection will occur 15 minutes past
        midnight everyday.";
    }
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default 120; // two minutes
        description
            "Specifies the maximum number of seconds that
            a NETCONF session may remain idle. A NETCONF
            session will be dropped if it is idle for an
            interval longer then this number of seconds.
            If set to zero, then the NETCONF client will
            never drop a session because it is idle.";
    }
    }
    }
    }
}
container reconnect-strategy {
    description
        "The reconnection strategy directs how a NETCONF client
        reconnects to a NETCONF server, after discovering its
        connection to the server has dropped, even if due to a
        reboot. The NETCONF client starts with the specified
        endpoint and tries to connect to it max-attempts times
        before trying the next endpoint in the list (round
        robin).";
    leaf start-with {
        type enumeration {
            enum first-listed {
                description
                    "Indicates that reconnections should start with
                    the first endpoint listed.";
            }
            enum last-connected {
                description
                    "Indicates that reconnections should start with
                    the endpoint last connected to. If no previous
                    connection has ever been established, then the

```



```

        first endpoint configured is used.  NETCONF
        clients SHOULD be able to remember the last
        endpoint connected to across reboots.";
    }
    enum random-selection {
        description
            "Indicates that reconnections should start with
            a random endpoint.";
    }
}
default "first-listed";
description
    "Specifies which of the NETCONF server's endpoints
    the NETCONF client should start with when trying
    to connect to the NETCONF server.";
}
leaf max-attempts {
    type uint8 {
        range "1..max";
    }
    default "3";
    description
        "Specifies the number times the NETCONF client tries
        to connect to a specific endpoint before moving on
        to the next endpoint in the list (round robin).";
}
}
} // netconf-server
} // initiate

container listen {
    if-feature "ssh-listen or tls-listen";
    presence "Enables client to accept call-home connections";
    description
        "Configures the client to accept call-home TCP connections.";
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default "3600"; // one hour
        description
            "Specifies the maximum number of seconds that a NETCONF
            session may remain idle. A NETCONF session will be
            dropped if it is idle for an interval longer than this
            number of seconds.  If set to zero, then the server
            will never drop a session because it is idle.  Sessions
            that have a notification subscription active are never
            dropped.";
    }
}
list endpoint {

```

```

    key "name";
    min-elements 1;
    description
        "List of endpoints to listen for NETCONF connections.";
    leaf name {
        type string;
        description
            "An arbitrary name for the NETCONF listen endpoint.";
    }
    uses netconf-client-listen-stack-grouping;
} // endpoint
} // listen
} // netconf-client-app-grouping

// Protocol accessible node, for servers that implement
// this module.
container netconf-client {
    uses netconf-client-app-grouping;
    description
        "Top-level container for NETCONF client configuration.";
}
}

```

<CODE ENDS>

3. The "ietf-netconf-server" Module

The NETCONF server model presented in this section supports both listening for connections as well as initiating call-home connections, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon parts of the model the NETCONF server supports.

3.1. Data Model Overview

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

3.1.1. Features

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

Features:

```
+-- ssh-listen
+-- tls-listen
+-- ssh-call-home
+-- tls-call-home
```

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

3.1.2. Groupings

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

Groupings:

```
+-- netconf-server-grouping
+-- netconf-server-listen-stack-grouping
+-- netconf-server-callhome-stack-grouping
+-- netconf-server-app-grouping
```

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

Each of these groupings are presented in the following subsections.

3.1.2.1. The "netconf-server-grouping" Grouping

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

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

```
grouping netconf-server-grouping
  +-- client-identity-mappings
      {(tls-listen or tls-call-home) and (sshcmn:ssh-x509-cert\
s)}}?
    +---u x509c2n:cert-to-name
```

Comments:

*The "netconf-server-grouping" defines the configuration for just "NETCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "SSH" or "TLS" protocol layers (for that, see [Section 3.1.2.2](#) and [Section 3.1.2.3](#)).

*The "client-identity-mappings" node, which must be enabled by "feature" statements, defines a mapping from certificate fields to NETCONF user names.

*For the referenced grouping statement(s):

-The "cert-to-name" grouping is discussed in [Section 4.1](#) of [[RFC7407](#)].

3.1.2.2. The "netconf-server-listen-stack-grouping" Grouping

The following tree diagram [[RFC8340](#)] illustrates the "netconf-server-listen-stack-grouping" grouping:

```

grouping netconf-server-listen-stack-grouping
+-- (transport)
+--:(ssh) {ssh-listen}?
|  +-- ssh
|    +-- tcp-server-parameters
|      | +---u tcps:tcp-server-grouping
|      +-- ssh-server-parameters
|        | +---u sshs:ssh-server-grouping
|        +-- netconf-server-parameters
|          +---u ncs:netconf-server-grouping
+--:(tls) {tls-listen}?
+-- tls
+-- tcp-server-parameters
|  +---u tcps:tcp-server-grouping
+-- tls-server-parameters
|  +---u tlss:tls-server-grouping
+-- netconf-server-parameters
+---u ncs:netconf-server-grouping

```

Comments:

*The "netconf-server-listen-stack-grouping" defines the configuration for a full NETCONF protocol stack for NETCONF servers that listen for standard connections from NETCONF clients, as opposed to initiating call-home [[RFC8071](#)] connections.

*The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

*For the referenced grouping statement(s):

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

3.1.2.3. The "netconf-server-callhome-stack-grouping" Grouping

The following tree diagram [[RFC8340](#)] illustrates the "netconf-server-callhome-stack-grouping" grouping:

```

grouping netconf-server-callhome-stack-grouping
  +-- (transport)
    +--:(ssh) {ssh-call-home}?
      |   +-- ssh
      |     +-- tcp-client-parameters
      |       |   +---u tcpc:tcp-client-grouping
      |       +-- ssh-server-parameters
      |         |   +---u sshs:ssh-server-grouping
      |         +-- netconf-server-parameters
      |           +---u ncs:netconf-server-grouping
    +--:(tls) {tls-call-home}?
      +-- tls
        +-- tcp-client-parameters
        |   +---u tcpc:tcp-client-grouping
        +-- tls-server-parameters
        |   +---u tlss:tls-server-grouping
        +-- netconf-server-parameters
            +---u ncs:netconf-server-grouping

```

Comments:

*The "netconf-server-callhome-stack-grouping" defines the configuration for a full NETCONF protocol stack, for NETCONF servers that initiate call-home [[RFC8071](#)] connections to NETCONF clients.

*The "transport" choice node enables both the SSH and TLS transports to be configured, with each option enabled by a "feature" statement.

*For the referenced grouping statement(s):

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

3.1.2.4. The "netconf-server-app-grouping" Grouping

The following tree diagram [[RFC8340](#)] illustrates the "netconf-server-app-grouping" grouping:

```

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

```

Comments:

*The "netconf-server-app-grouping" defines the configuration for a NETCONF server that supports both listening for connections from NETCONF clients as well as 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>
        </tls-server-parameters>
      </tls>
    </endpoint>
  </listen>
</netconf-server>
```

```

        </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-address>
                    </tcp-client-parameters>
                    <keepalives>
                        <idle-time>15</idle-time>
                        <max-probes>3</max-probes>
                        <probe-interval>30</probe-interval>
                    </keepalives>
                </ssh>
            </endpoint>
        </endpoints>
    </netconf-client>

```

```

    </tcp-client-parameters>
    <ssh-server-parameters>
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <keystore-reference>ssh-rsa-key</keystore-refere\
nce>
          </public-key>
        </host-key>
      </server-identity>
      <client-authentication>
        <supported-authentication-methods>
          <publickey/>
        </supported-authentication-methods>
      </client-authentication>
    </ssh-server-parameters>
    <netconf-server-parameters>
      <!-- nothing to configure -->
    </netconf-server-parameters>
  </ssh>
</endpoint>
<endpoint>
  <name>west-data-center</name>
  <ssh>
    <tcp-client-parameters>
      <remote-address>west.config-mgr.example.com</remote-ad\
dress>
    </tcp-client-parameters>
    <ssh-server-parameters>
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <keystore-reference>ssh-rsa-key</keystore-refere\
nce>
          </public-key>
        </host-key>
      </server-identity>
      <client-authentication>
        <supported-authentication-methods>
          <publickey/>
        </supported-authentication-methods>
      </client-authentication>
    </ssh-server-parameters>
    <netconf-server-parameters>
      <!-- nothing to configure -->
    </netconf-server-parameters>
  </ssh>

```

```

        </endpoint>
    </endpoints>
    <connection-type>
        <periodic>
            <idle-timeout>300</idle-timeout>
            <period>60</period>
        </periodic>
    </connection-type>
    <reconnect-strategy>
        <start-with>last-connected</start-with>
        <max-attempts>3</max-attempts>
    </reconnect-strategy>
</netconf-client>
<netconf-client> <!-- TLS-based client -->
    <name>data-collector</name>
    <endpoints>
        <endpoint>
            <name>east-data-center</name>
            <tls>
                <tcp-client-parameters>
                    <remote-address>east.analytics.example.com</remote-add\
ress>

                    <keepalives>
                        <idle-time>15</idle-time>
                        <max-probes>3</max-probes>
                        <probe-interval>30</probe-interval>
                    </keepalives>
                </tcp-client-parameters>
                <tls-server-parameters>
                    <server-identity>
                        <certificate>
                            <keystore-reference>
                                <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>

                                <certificate>ex-rsa-cert</certificate>
                            </keystore-reference>
                        </certificate>
                    </server-identity>
                    <client-authentication>
                        <ca-certs>
                            <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>

                            </ca-certs>
                            <ee-certs>
                                <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>

                                </ee-certs>
                            </client-authentication>
                        <keepalives>

```

```

        <test-peer-aliveness>
            <max-wait>30</max-wait>
            <max-attempts>3</max-attempts>
        </test-peer-aliveness>
    </keepalives>
</tls-server-parameters>
<netconf-server-parameters>
    <client-identity-mappings>
        <cert-to-name>
            <id>1</id>
            <fingerprint>11:0A:05:11:00</fingerprint>
            <map-type>x509c2n:specified</map-type>
            <name>scooby-doo</name>
        </cert-to-name>
        <cert-to-name>
            <id>2</id>
            <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
    </client-identity-mappings>
</netconf-server-parameters>
</tls>
</endpoint>
<endpoint>
    <name>west-data-center</name>
    <tls>
        <tcp-client-parameters>
            <remote-address>west.analytics.example.com</remote-add\
ress>

            <keepalives>
                <idle-time>15</idle-time>
                <max-probes>3</max-probes>
                <probe-interval>30</probe-interval>
            </keepalives>
        </tcp-client-parameters>
        <tls-server-parameters>
            <server-identity>
                <certificate>
                    <keystore-reference>
                        <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>

                        <certificate>ex-rsa-cert</certificate>
                    </keystore-reference>
                </certificate>
            </server-identity>
            <client-authentication>
                <ca-certs>
                    <truststore-reference>trusted-client-ca-certs</tru\
ststore-reference>

                </ca-certs>

```

```

        <ee-certs>
            <truststore-reference>trusted-client-ee-certs</tru\
ststore-reference>
        </ee-certs>
    </client-authentication>
    <keepalives>
        <test-peer-aliveness>
            <max-wait>30</max-wait>
            <max-attempts>3</max-attempts>
        </test-peer-aliveness>
    </keepalives>
</tls-server-parameters>
<netconf-server-parameters>
    <client-identity-mappings>
        <cert-to-name>
            <id>1</id>
            <fingerprint>11:0A:05:11:00</fingerprint>
            <map-type>x509c2n:specified</map-type>
            <name>scooby-doo</name>
        </cert-to-name>
        <cert-to-name>
            <id>2</id>
            <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
    </client-identity-mappings>
</netconf-server-parameters>
</tls>
</endpoint>
</endpoints>
<connection-type>
    <persistent/>
</connection-type>
<reconnect-strategy>
    <start-with>first-listed</start-with>
    <max-attempts>3</max-attempts>
</reconnect-strategy>
</netconf-client>
</call-home>
</netconf-server>

```

3.3. YANG Module

This YANG module has normative references to [[RFC6242](#)], [[RFC6991](#)], [[RFC7407](#)], [[RFC7589](#)], [[RFC8071](#)], [[I-D.ietf-netconf-tcp-client-server](#)], [[I-D.ietf-netconf-ssh-client-server](#)], and [[I-D.ietf-netconf-tls-client-server](#)].

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

```

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

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

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

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

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

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

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

  import ietf-tls-server {
    prefix tlss;
    revision-date 2020-08-20; // stable grouping definitions
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }
}

```


organization

"IETF NETCONF (Network Configuration) Working Group";

contact

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

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

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

Author: Gary Wu <<mailto:garywu@cisco.com>>

Author: Juergen Schoenwaelder

<<mailto:j.schoenwaelder@jacobs-university.de>>;

description

"This module contains a collection of YANG definitions
for configuring NETCONF servers.

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

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

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

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

revision 2020-08-20 {

description

"Initial version";

reference

"RFC HHHH: NETCONF Client and Server Models";

}

// Features

feature ssh-listen {

description

"The 'ssh-listen' feature indicates that the NETCONF server
supports opening a port to accept NETCONF over SSH
client connections.";

```

reference
  "RFC 6242:
    Using the NETCONF Protocol over Secure Shell (SSH)";
}

feature tls-listen {
  description
    "The 'tls-listen' feature indicates that the NETCONF server
    supports opening a port to accept NETCONF over TLS
    client connections.";
  reference
    "RFC 7589: Using the NETCONF Protocol over Transport
    Layer Security (TLS) with Mutual X.509
    Authentication";
}

feature ssh-call-home {
  description
    "The 'ssh-call-home' feature indicates that the NETCONF
    server supports initiating a NETCONF over SSH call
    home connection to NETCONF clients.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

feature tls-call-home {
  description
    "The 'tls-call-home' feature indicates that the NETCONF
    server supports initiating a NETCONF over TLS call
    home connection to NETCONF clients.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

// Groupings

grouping netconf-server-grouping {
  description
    "A reusable grouping for configuring a NETCONF server
    without any consideration for how underlying transport
    sessions are established.

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

```

```

    to consuming models.";

container client-identity-mappings {
  if-feature
    "(tls-listen or tls-call-home) and (sshcmn:ssh-x509-certs)";
  description
    "Specifies mappings through which NETCONF client X.509
    certificates are used to determine a NETCONF username.
    If no matching and valid cert-to-name list entry can be
    found, then the NETCONF server MUST close the connection,
    and MUST NOT accept NETCONF messages over it.";
  reference
    "RFC 7407: A YANG Data Model for SNMP Configuration.";
  uses x509c2n:cert-to-name {
    refine "cert-to-name/fingerprint" {
      mandatory false;
      description
        "A 'fingerprint' value does not need to be specified
        when the 'cert-to-name' mapping is independent of
        fingerprint matching. A 'cert-to-name' having no
        fingerprint value will match any client certificate
        and therefore should only be present at the end of
        the user-ordered 'cert-to-name' list.";
    }
  }
}

}

}

grouping netconf-server-listen-stack-grouping {
  description
    "A reusable grouping for configuring a NETCONF server
    'listen' protocol stack for a single connection.";
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";
    case ssh {
      if-feature "ssh-listen";
      container ssh {
        description
          "SSH-specific listening configuration for inbound
          connections.";
        container tcp-server-parameters {
          description
            "A wrapper around the TCP client parameters
            to avoid name collisions.";
          uses tcps:tcp-server-grouping {
            refine "local-port" {
              default "830";
            }
          }
        }
      }
    }
  }
}

```

```

        description
            "The NETCONF server will listen on the
            IANA-assigned well-known port value
            for 'netconf-ssh' (830) if no value
            is specified.";
    }
}
}
container ssh-server-parameters {
    description
        "A wrapper around the SSH server parameters
        to avoid name collisions.";
    uses sshs:ssh-server-grouping;
}
container netconf-server-parameters {
    description
        "A wrapper around the NETCONF server parameters
        to avoid name collisions.";
    uses ncs:netconf-server-grouping;
}
}
}
case tls {
    if-feature "tls-listen";
    container tls {
        description
            "TLS-specific listening configuration for inbound
            connections.";
        container tcp-server-parameters {
            description
                "A wrapper around the TCP client parameters
                to avoid name collisions.";
            uses tcps:tcp-server-grouping {
                refine "local-port" {
                    default "6513";
                    description
                        "The NETCONF server will listen on the
                        IANA-assigned well-known port value
                        for 'netconf-tls' (6513) if no value
                        is specified.";
                }
            }
        }
    }
}
container tls-server-parameters {
    description
        "A wrapper around the TLS server parameters to
        avoid name collisions.";
    uses tlss:tls-server-grouping {
        refine "client-authentication" {

```

```

        must 'ca-certs or ee-certs';
        description
            "NETCONF/TLS servers MUST validate client
            certificates. This configures certificates
            at the socket-level (i.e. bags), more
            discriminating client-certificate checks
            SHOULD be implemented by the application.";
        reference
            "RFC 7589:
            Using the NETCONF Protocol over Transport Layer
            Security (TLS) with Mutual X.509 Authentication";
    }
}
}
container netconf-server-parameters {
    description
        "A wrapper around the NETCONF server parameters
        to avoid name collisions.";
    uses ncs:netconf-server-grouping;
}
}
}
}
}

```

```

grouping netconf-server-callhome-stack-grouping {
    description
        "A reusable grouping for configuring a NETCONF server
        'call-home' protocol stack, for a single connection.";
    choice transport {
        mandatory true;
        description
            "Selects between available transports.";
        case ssh {
            if-feature "ssh-call-home";
            container ssh {
                description
                    "Specifies SSH-specific call-home transport
                    configuration.";
                container tcp-client-parameters {
                    description
                        "A wrapper around the TCP client parameters
                        to avoid name collisions.";
                    uses tcpc:tcp-client-grouping {
                        refine "remote-port" {
                            default "4334";
                            description
                                "The NETCONF server will attempt to connect
                                to the IANA-assigned well-known port for

```

```

        'netconf-ch-tls' (4334) if no value is
        specified.";
    }
}
}
container ssh-server-parameters {
    description
        "A wrapper around the SSH server parameters
        to avoid name collisions.";
    uses sshs:ssh-server-grouping;
}
container netconf-server-parameters {
    description
        "A wrapper around the NETCONF server parameters
        to avoid name collisions.";
    uses ncs:netconf-server-grouping;
}
}
}
case tls {
    if-feature "tls-call-home";
    container tls {
        description
            "Specifies TLS-specific call-home transport
            configuration.";
        container tcp-client-parameters {
            description
                "A wrapper around the TCP client parameters
                to avoid name collisions.";
            uses tcpc:tcp-client-grouping {
                refine "remote-port" {
                    default "4335";
                    description
                        "The NETCONF server will attempt to connect
                        to the IANA-assigned well-known port for
                        'netconf-ch-tls' (4335) if no value is
                        specified.";
                }
            }
        }
    }
}
}
container tls-server-parameters {
    description
        "A wrapper around the TLS server parameters to
        avoid name collisions.";
    uses tlss:tls-server-grouping {
        refine "client-authentication" {
            must 'ca-certs or ee-certs';
            description
                "NETCONF/TLS servers MUST validate client

```

```

        certificates. This configures certificates
        at the socket-level (i.e. bags), more
        discriminating client-certificate checks
        SHOULD be implemented by the application.";
    reference
        "RFC 7589:
        Using the NETCONF Protocol over Transport Layer
        Security (TLS) with Mutual X.509 Authentication";
    }
}
}
}
}
}
}
}
}
}
}

grouping netconf-server-app-grouping {
    description
        "A reusable grouping for configuring a NETCONF server
        application that supports both 'listen' and 'call-home'
        protocol stacks for a multiplicity of connections.";
    container listen {
        if-feature "ssh-listen or tls-listen";
        presence
            "Enables server to listen for NETCONF client connections.";
        description
            "Configures listen behavior";
        leaf idle-timeout {
            type uint16;
            units "seconds";
            default 3600; // one hour
            description
                "Specifies the maximum number of seconds that a NETCONF
                session may remain idle. A NETCONF session will be
                dropped if it is idle for an interval longer than this
                number of seconds. If set to zero, then the server
                will never drop a session because it is idle. Sessions
                that have a notification subscription active are never
                dropped.";
        }
        list endpoint {
            key "name";
            min-elements 1;

```

```

    description
        "List of endpoints to listen for NETCONF connections.";
    leaf name {
        type string;
        description
            "An arbitrary name for the NETCONF listen endpoint.";
    }
    uses netconf-server-listen-stack-grouping;
}
}
container call-home {
    if-feature "ssh-call-home or tls-call-home";
    presence
        "Enables the NETCONF server to initiate the underlying
        transport connection to NETCONF clients.";
    description "Configures call home behavior.";
    list netconf-client {
        key "name";
        min-elements 1;
        description
            "List of NETCONF clients the NETCONF server is to
            maintain simultaneous call-home connections with.";
        leaf name {
            type string;
            description
                "An arbitrary name for the remote NETCONF client.";
        }
    }
    container endpoints {
        description
            "Container for the list of endpoints.";
        list endpoint {
            key "name";
            min-elements 1;
            ordered-by user;
            description
                "A non-empty user-ordered list of endpoints for this
                NETCONF server to try to connect to in sequence.
                Defining more than one enables high-availability.";
            leaf name {
                type string;
                description
                    "An arbitrary name for this endpoint.";
            }
            uses netconf-server-callhome-stack-grouping;
        }
    }
}
container connection-type {
    description
        "Indicates the NETCONF server's preference for how the

```



```

    NETCONF connection is maintained.";
choice connection-type {
  mandatory true;
  description
    "Selects between available connection types.";
  case persistent-connection {
    container persistent {
      presence "Indicates that a persistent connection is
        to be maintained.";
      description
        "Maintain a persistent connection to the NETCONF
          client. If the connection goes down, immediately
          start trying to reconnect to the NETCONF client,
          using the reconnection strategy.

          This connection type minimizes any NETCONF client
          to NETCONF server data-transfer delay, albeit at
          the expense of holding resources longer.";
    }
  }
  case periodic-connection {
    container periodic {
      presence "Indicates that a periodic connection is
        to be maintained.";
      description
        "Periodically connect to the NETCONF client.

        This connection type increases resource
        utilization, albeit with increased delay in
        NETCONF client to NETCONF client interactions.

        The NETCONF client SHOULD gracefully close the
        connection using <close-session> upon completing
        planned activities. If the NETCONF session is
        not closed gracefully, the NETCONF server MUST
        immediately attempt to reestablish the connection.

        In the case that the previous connection is still
        active (i.e., the NETCONF client has not closed
        it yet), establishing a new connection is NOT
        RECOMMENDED.";
      leaf period {
        type uint16;
        units "minutes";
        default "60";
        description
          "Duration of time between periodic connections.";
      }
      leaf anchor-time {

```

```

    type yang:date-and-time {
        // constrained to minute-level granularity
        pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
            + '(Z|[\+\-]\d{2}:\d{2})';
    }
    description
        "Designates a timestamp before or after which a
        series of periodic connections are determined.
        The periodic connections occur at a whole
        multiple interval from the anchor time. For
        example, for an anchor time is 15 minutes past
        midnight and a period interval of 24 hours, then
        a periodic connection will occur 15 minutes past
        midnight everyday.";
}
leaf idle-timeout {
    type uint16;
    units "seconds";
    default 120; // two minutes
    description
        "Specifies the maximum number of seconds that
        a NETCONF session may remain idle. A NETCONF
        session will be dropped if it is idle for an
        interval longer than this number of seconds.
        If set to zero, then the server will never
        drop a session because it is idle.";
}
}
} // case periodic-connection
} // choice connection-type
} // container connection-type
container reconnect-strategy {
    description
        "The reconnection strategy directs how a NETCONF server
        reconnects to a NETCONF client, after discovering its
        connection to the client has dropped, even if due to a
        reboot. The NETCONF server starts with the specified
        endpoint and tries to connect to it max-attempts times
        before trying the next endpoint in the list (round
        robin).";
    leaf start-with {
        type enumeration {
            enum first-listed {
                description
                    "Indicates that reconnections should start with
                    the first endpoint listed.";
            }
            enum last-connected {
                description

```

```

        "Indicates that reconnections should start with
        the endpoint last connected to. If no previous
        connection has ever been established, then the
        first endpoint configured is used. NETCONF
        servers SHOULD be able to remember the last
        endpoint connected to across reboots.";
    }
    enum random-selection {
        description
            "Indicates that reconnections should start with
            a random endpoint.";
    }
}
default "first-listed";
description
    "Specifies which of the NETCONF client's endpoints
    the NETCONF server should start with when trying
    to connect to the NETCONF client.";
}
leaf max-attempts {
    type uint8 {
        range "1..max";
    }
    default "3";
    description
        "Specifies the number times the NETCONF server tries
        to connect to a specific endpoint before moving on
        to the next endpoint in the list (round robin).";
}
} // container reconnect-strategy
} // list netconf-client
} // container call-home
} // grouping netconf-server-app-grouping

// Protocol accessible node, for servers that implement
// this module.
container netconf-server {
    uses netconf-server-app-grouping;
    description
        "Top-level container for NETCONF server configuration.";
}
}

```

<CODE ENDS>

4. Security Considerations

4.1. The "ietf-netconf-client" YANG Module

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

The NETCONF access control model (NACM) [[RFC8341](#)] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

4.2. The "ietf-netconf-server" YANG Module

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

The NETCONF access control model (NACM) [[RFC8341](#)] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

Please be aware that this module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

5. IANA Considerations

5.1. The "IETF XML" Registry

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

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-client

Registrant Contact: The NETCONF WG of the IETF.

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

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-server

Registrant Contact: The NETCONF WG of the IETF.

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

5.2. The "YANG Module Names" Registry

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

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

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

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore] Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-

netconf-keystore-19, 10 July 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-keystore-19>>.

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

Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-21, 10 July 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>>.

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

Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-07, 8 July 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>>.

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

Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-21, 10 July 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

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

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

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

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

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <<https://www.rfc-editor.org/info/rfc7407>>.

[RFC7589]

Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <<https://www.rfc-editor.org/info/rfc7589>>.

[RFC7950]

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

[RFC8174]

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

6.2. Informative References

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

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

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

Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-04, 8 July 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>>.

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

Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-20, 8 July 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>>.

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

Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-20, 8 July 2020, <<https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>>.

[I-D.ietf-netconf-trust-anchors]

Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-

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

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<https://www.rfc-editor.org/info/rfc8071>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.

Appendix A. Change Log

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

A.1. 00 to 01

*Renamed "keychain" to "keystore".

A.2. 01 to 02

*Added to ietf-netconf-client ability to connected to a cluster of endpoints, including a reconnection-strategy.

*Added to ietf-netconf-client the ability to configure connection-type and also keep-alive strategy.

*Updated both modules to accommodate new groupings in the ssh/tls drafts.

A.3. 02 to 03

*Refined use of `tls-client-grouping` to add a `must` statement indicating that the TLS client must specify a `client-certificate`.

*Changed `'netconf-client'` to be a grouping (not a container).

A.4. 03 to 04

*Added RFC 8174 to Requirements Language Section.

*Replaced `refine` statement in `ietf-netconf-client` to add a mandatory `true`.

*Added `refine` statement in `ietf-netconf-server` to add a `must` statement.

*Now there are containers and groupings, for both the client and server models.

A.5. 04 to 05

*Now tree diagrams reference `ietf-netmod-yang-tree-diagrams`

*Updated examples to inline key and certificates (no longer a `leafref` to `keystore`)

A.6. 05 to 06

*Fixed change log missing section issue.

*Updated examples to match latest updates to the `crypto-types`, `trust-anchors`, and `keystore` drafts.

*Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

*Removed `"idle-timeout"` from `"persistent"` connection config.

*Added `"random-selection"` for `reconnection-strategy's "starts-with"` enum.

*Replaced `"connection-type"` choice default (`persistent`) with `"mandatory true"`.

*Reduced the `periodic-connection's "idle-timeout"` from 5 to 2 minutes.

*Replaced `reconnect-timeout` with `period/anchor-time` combo.

A.8. 07 to 08

- *Modified examples to be compatible with new crypto-types algs

A.9. 08 to 09

- *Corrected use of "mandatory true" for "address" leafs.

- *Updated examples to reflect update to groupings defined in the keystore draft.

- *Updated to use groupings defined in new TCP and HTTP drafts.

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

A.10. 09 to 10

- *Reformatted YANG modules.

A.11. 10 to 11

- *Adjusted for the top-level "demux container" added to groupings imported from other modules.

- *Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

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

- *Moved "expanded" tree diagrams to the Appendix.

A.12. 11 to 12

- *Removed the "Design Considerations" section.

- *Removed the 'must' statement limiting keepalives in periodic connections.

- *Updated models and examples to reflect removal of the "demux" containers in the imported models.

- *Updated the "periodic-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.

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