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

This document contains references to other drafts in progress, both in the Normative References section, as well as in body text throughout. Please update the following references to reflect their final RFC assignments:

- I-D.ietf-netconf-keystore
- I-D.ietf-netconf-tcp-client-server
- I-D.ietf-netconf-ssh-client-server
- I-D.ietf-netconf-tls-client-server

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

- "XXXX" --> the assigned RFC value for this draft
- "AAAA" --> the assigned RFC value for I-D.ietf-netconf-tcp-client-server
- "YYYY" --> the assigned RFC value for I-D.ietf-netconf-ssh-client-server
Internet-Draft       NETCONF Client and Server Models       November 2019

- "ZZZZ" --> the assigned RFC value for I-D.ietf-netconf-tls-client-server

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

- "2019-11-20" --> the publication date of this draft

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

- Appendix B. Change Log

Status of This Memo

This Internet-Draft 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 23, 2020.

Copyright Notice

Copyright (c) 2019 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.
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].

2. 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.

3. The NETCONF Client Model

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.

3.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-client" module.

This tree diagram only shows the nodes defined in this module; it does show the nodes defined by "grouping" statements used by this module.

Please see Appendix A.1 for a tree diagram that illustrates what the module looks like with all the "grouping" statements expanded.

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

   grouping netconf-client-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
```
### 3.2. Example Usage

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

This example is 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].

```xml
<netconf-client
 xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-client"
 <!-- 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>
                 <local-definition>
                   <algorithm>rsa2048</algorithm>
                   <public-key-format>ct:ssh-public-key-format</public-key-format>
                 </local-definition>
                 <public-key>base64encodedvalue==</public-key>
                 <private-key-format>ct:rsa-private-key-format</private-key-format>
                 <private-key>base64encodedvalue==</private-key>
               </client-identity>
             </ssh-client-parameters>
           </ssh>
         </endpoint>
       </endpoints>
   </netconf-server>
 </initiate>
</netconf-client>
```
<server-authentication>
<ca-certs>
<truststore-reference>explicitly-trusted-server-ca\-certs</truststore-reference>
</ca-certs>
<server-certs>
<truststore-reference>explicitly-trusted-server-certs</truststore-reference>
</server-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>
<local-definition>
<algorithm>rsa2048</algorithm>
</local-definition>
</certificate>
</client-identity>
<server-authentication>
<ca-certs>
<truststore-reference>explicitly-trusted-server-ca\</truststore-reference>
  <ca-certs>
    <server-certs>
      <truststore-reference>explicitly-trusted-server-certs</truststore-reference>
    </server-certs>
  </ca-certs>
</server-authentication>
<keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</keepalives>
</tls-client-parameters>
<netconf-client-parameters>
  <!-- nothing to configure -->
</netconf-client-parameters>
</tls>
</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 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>
            <local-definition>
              <algorithm>rsa2048</algorithm>
              <public-key-format>ct:ssh-public-key-format</public-key-format>
            </local-definition>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
          <private-key-format>ct:rsa-private-key-format</private-key-format>
          <private-key>base64encodedvalue==</private-key>
        </client-identity>
      </ssh-client-parameters>
    </ssh>
  </endpoint>
</listen>
3.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7589], [RFC8071], [I-D.kwatsen-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@2019-11-20.yang"

module ietf-netconf-client {
  yang-version 1.1;
  prefix ncc;

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

  import ietf-tcp-client {
    prefix tcpc;


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

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 (https://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX
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 2019-11-20 {
  description
    "Initial version";
  reference
    "RFC XXXX: 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 tcp: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' (443) 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
grouping netconf-client-listen-stack-grouping {
    description "A reusable grouping for configuring a NETCONF client '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 server parameters to avoid name collisions.";
                    uses tcpss: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 sshs:ssh-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
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 from the list."
      }
    }
  }
```
"Indicates that reconnections should start with a random endpoint."

} // netconf-server

} // initiate

container listen {
  if-feature "ssh-listen or tls-listen";
  presence "Enables client to accept call-home connections";
  description
    "Configures client accepting 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 {

4. The NETCONF Server Model

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.

4.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-server" module.

This tree diagram only shows the nodes defined in this module; it does show the nodes defined by "grouping" statements used by this module.

Please see Appendix A.2 for a tree diagram that illustrates what the module looks like with all the "grouping" statements expanded.
+- client-identity-mappings
  | {tls-listen or tls-call-home or sshcmn:ssh-x509-certs)?
  |   +--> x509c2n:cert-to-name
  | grouping netconf-server-listen-stack-grouping
  +- (transport)
      +--:(ssh) {ssh-listen}?
          +- ssh
          | +- tcp-server-parameters
          |   | +- tcp:tcp-server-grouping
          | +- ssh-server-parameters
          |   | +- sshs:ssh-server-grouping
          | +- netconf-server-parameters
          |   | +- ncs:netconf-server-grouping
          +-:(tls) {tls-listen}?
              +- tls
              | +- tcp-server-parameters
              |   | +- tcp:tcp-server-grouping
              | +- tls-server-parameters
              |   | +- tlss:tls-server-grouping
              | +- netconf-server-parameters
              |   | +- ncs:netconf-server-grouping
      grouping netconf-server-callhome-stack-grouping
      +- (transport)
          +--:(ssh) {ssh-call-home}?
              +- ssh
              | +- tcp-client-parameters
              |   | +- tcp:tcp-client-grouping
              | +- ssh-server-parameters
              |   | +- sshs:ssh-server-grouping
              | +- netconf-server-parameters
              |   | +- ncs:netconf-server-grouping
          +--:(tls) {tls-call-home}?
              +- tls
              | +- tcp-client-parameters
              |   | +- tcp:tcp-client-grouping
              | +- tls-server-parameters
              |   | +- tlss:tls-server-grouping
              | +- netconf-server-parameters
              |   | +- ncs:netconf-server-grouping
          grouping netconf-server-app-grouping
          +- listen! {ssh-listen or tls-listen}?
              | +- idle-timeout? uint16
              | +- endpoint* [name]
              |   | +- name? string
              |   | +- netconf-server-listen-stack-grouping
          +- call-home! {ssh-call-home or tls-call-home}?
              +- netconf-client* [name]
              | +- name? string

Watsen                    Expires May 23, 2020                 [Page 21]
4.2. 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 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

<netconf-server
xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server"
xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">

<!-- endpoints to listen for NETCONF connections on -->
<listen>
  <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>
          </host-key>
        </server-identity>
      </ssh-server-parameters>
    </ssh>
  </endpoint>
</listen>
<local-definition>
  <algorithm>rsa2048</algorithm>
  <public-key-format>ct:ssh-public-key-format</public-key-format>
  <host-key>
    <public-key>
      base64encodedvalue==
    </public-key>
    <private-key>
      base64encodedvalue==
    </private-key>
  </host-key>
</local-definition>

<server-identity>
  <client-authentication>
    <supported-authentication-methods>
      <public-key/>
    </supported-authentication-methods>
  </client-authentication>
</server-identity>

<client-authentication>
  <ca-certs>
    <truststore-reference>explicitly-trusted-client-ca-certificates</truststore-reference>
  </ca-certs>
</client-authentication>

<netconf-server-parameters>
  <!-- nothing to configure -->
</netconf-server-parameters>
</ssh>

<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>
</endpoint>
</ca-certs>
</client-certs>
	<truststore-reference>explicitly-trusted-client-certs</truststore-reference>
</client-authentication>
</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>
	</ssh>
	</endpoint>
</endpoints>
</netconf-client>
</call-home>
<private-key-format>
  <private-key>base64encodedvalue==</private-key>
</local-definition>
</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-address>
    </tcp-client-parameters>
  </ssh>
</endpoint>
<server-identity>
  <host-key>
    <name>deployment-specific-certificate</name>
    <public-key>
      <local-definition>
        <algorithm>rsa2048</algorithm>
        <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>
        <private-key>base64encodedvalue==</private-key>
      </local-definition>
    </public-key>
  </host-key>
</server-identity>
<client-authentication>
  <supported-authentication-methods>
    <publickey/>
  </supported-authentication-methods>
</client-authentication>
</ssh-server-parameters>
<netconf-server-parameters>
</netconf-server-parameters>
<!-- nothing to configure -->
</netconf-server-parameters>
</ssh>
</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-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>
<local-definition>
<algorithm>rsa2048</algorithm>
</local-definition>
</certificate>
</server-identity>
</client-authentication>
<ca-certs>
<client-certs>
  <truststore-reference>explicitly-trusted-client-ca\-certs</truststore-reference>
  </ca-certs>
</client-authentication>
<keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</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>
        <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>
    <cert-to-name>
      <id>2</id>
      <map-type>x509c2n:san-any</map-type>
      <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>
    </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-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>
          <local-definition>
            <algorithm>rsa2048</algorithm>
            <public-key-format>ct:subject-public-key-info-format</public-key-format>
          </certificate>
        </server-identity>
      </tls-server-parameters>
    </tls>
  </tls>
</endpoint>
<public-key>base64encodedvalue==</public-key>
/private-key-format
<private-key>base64encodedvalue==</private-key>
<cert>base64encodedvalue==</cert>
</local-definition>
</certificate>
</server-identity>
</client-authentication>
<ca-certs>
<truststore-reference>explicitly-trusted-client-ca\-certs</truststore-reference>
</ca-certs>
</client-certs>
</truststore-reference>
</client-certs>
</client-authentication>
<keepalives>
<max-wait>30</max-wait>
<max-attempts>3</max-attempts>
</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:sn-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>
4.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7407], [RFC7589], [RFC8071], [I-D.kwatsen-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@2019-11-20.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 BBBB: YANG Groupings for TCP Clients and TCP Servers";
    }

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

    import ietf-ssh-common {
        prefix sshcmn;
        revision-date 2019-11-20; // stable grouping definitions
        reference
            "RFC CCCC: YANG Groupings for SSH Clients and SSH Servers";
    }

</CODE BEGINS>
import ietf-ssh-server {
    prefix sshs;
    revision-date 2019-11-20; // stable grouping definitions
    reference "RFC CCCC: YANG Groupings for SSH Clients and SSH Servers";
}

import ietf-tls-server {
    prefix tlss;
    revision-date 2019-11-20; // stable grouping definitions
    reference "RFC DDDD: YANG Groupings for TLS Clients and TLS Servers";
}

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

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

    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 (https://trustee.ietf.org/license-info).

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

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

revision 2019-11-20 {
  description
    "Initial version";
  reference
    "RFC XXXX: 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 or 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

  Watsen                    Expires May 23, 2020                 [Page 32]
'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.";
          }
        }
      }
    }
  }
  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 "830";
            description "The NETCONF server will listen on the IANA-assigned well-known port value for 'netconf-tls' (830) if no value is specified.";
          }
        }
      }
    }
  }
}
"A wrapper around the TCP client parameters to avoid name collisions.";
uses tcp: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 tls:tls-server-grouping; /* {
    FIXME: commented out since auth could also be external.
    ^-- need a better ‘must’ expression?
    refine "client-authentication" {
      must ‘ca-certs or client-certs’;
      description
      "NETCONF/TLS servers MUST validate client certificates.";
    }
  }*/
}

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.";
                }
            }
        }
    }
}
"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; /* {
    FIXME: commented out since auth could also be external.
    ^-- need a better 'must' expression?
    refine "client-authentication" {
      must 'ca-certs or client-certs';
      description
        "NETCONF/TLS servers MUST validate client
         certificates.";
    }
  }*/
}

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
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
// Protocol accessible node, for servers that implement this
// module.

carrier netconf-server {
    uses netconf-server-app-grouping;
    description
        "Top-level container for NETCONF server configuration.";
}

5. Security Considerations

The YANG module defined in this document uses groupings defined in
[I-D.kwatsen-netconf-tcp-client-server],
[I-D.ietf-netconf-ssh-client-server], and
[I-D.ietf-netconf-tls-client-server]. Please see the Security
Considerations section in those documents for concerns related those
groupings.

The YANG modules defined in this document 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.

There are a number of data nodes defined in the YANG modules that are
writable/creatable/deletable (i.e., config true, which is the
default). Some of 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:

None of the subtrees or data nodes in the modules defined in this
document need to be protected from write operations.

Some of the readable data nodes in the YANG modules may be considered
sensitive or vulnerable in some network environments. It is thus
important to control read access (e.g., via get, get-config, or
notification) to these data nodes. These are the subtrees and data
nodes and their sensitivity/vulnerability:
None of the subtrees or data nodes in the modules defined in this document need to be protected from read operations.

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

The modules defined in this document do not define any 'RPC' or 'action' statements.

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:

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

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

6.2. The YANG Module Names Registry

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

- name: ietf-netconf-client
  prefix: ncc
  reference: RFC XXXX

- name: ietf-netconf-server
  prefix: ncs
  reference: RFC XXXX

7. References
7.1. Normative References

[I-D.ietf-netconf-keystore]

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

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

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


7.2. Informative References

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


Appendix A. Expanded Tree Diagrams

A.1. Expanded Tree Diagram for 'ietf-netconf-client'

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-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 3.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

========== NOTE: `\` line wrapping per BCP XXX (RFC XXXX) =========
Watsen                    Expires May 23, 2020                 [Page 50]
|     |              |  |     |        |     +--rw cert? |              |  |     |        |     |       end-entity\ |              |  |     |        |     |       +--rw certificate-\ |  
|     |              |  |     |        |     +---n certificate-\ |              |  |     |        |     |          yang:da\ |              |  |     |        |     |          +-- expiration-\ |  
|     |              |  |     |        |     +---x generate-cer\ |              |  |     |        |     |          bina\ |              |  |     |        |     |          +----w input |              |  |     |        |     |          +----w subject |  
|     |              |  |     |        |     +----w attrib |              |  |     |        |     |          bina\ |              |  |     |        |     |          +--ro output |              |  |     |        |     |          +--ro certificate |  
|     |              |  |     |        |     +--:(keystore) |              |  |     |        |     |          {keystore-supported}\ |              |  |     |        |     |          +--rw keystore-reference |  
|     |              |  |     |        |     +--rw asymmetric-key |              |  |     |        |     |          {keystore-defined}\ |              |  |     |        |     |          +--rw asymmetric-key |  
|     |              |  |     |        |     +--rw certificate-key |              |  |     |        |     |          +--rw raw-public-key |              |  |     |        |     |          +--rw raw-public-key |  
|     |              |  |     |        |     +--rw (local-or-keystore) |              |  |     |        |     |          +--:(local) |              |  |     |        |     |          +--rw local-definition |  
|     |              |  |     |        |     +--rw algorithm |              |  |     |        |     |          iasa:asymmetric |  

Watsen                    Expires May 23, 2020                 

[Page 52]
Internet-Draft      NETCONF Client and Server Models       November 2019

|     |              |  |     |        |     +--rw public-key-format?
|     |              |  |     |        |     |       identityref
|     |              |  |     |        |     +--rw public-key
|     |              |  |     |        |     |       binary
|     |              |  |     |        |     +--rw private-key-
\format?
|     |              |  |     |        |     |       identityref
|     |              |  |     |        |     +--rw (private-key
\-type)
|     |              |  |     |        |     |       empty
\y
|     |              |  |     |        |     +--: (private-key
\e-key?
|     |              |  |     |        |     |       binary
\ry
|     |              |  |     |        |     +--: (hidden-private
\vate-key)
\-private-key?
|     |              |  |     |        |     |       empty
\private-key)
|     |              |  |     |        |     +--: (encrypted-
\ted-private-key
\y-type)
\-symmetric-key-ref)
\rw symmetric-key-ref? leafref
\ (keystore-supported)?
\-symmetric-key-ref)
\rw asymmetric-key-ref? leafref
\ (keystore-supported)?
\ue?
\inary
\rted)?
\nce?

Watsen                    Expires May 23, 2020                 [Page 53]
ks:asymmetric\n
| key-ref | +---:(psk)
|        |   +---rw psk {psk-auth}?
|        |     +---rw (local-or-keystore)
|        |       +---:(local)
|        |         | {local-definition}
| ons-supported)?
|        |     +---rw local-definition
|        |           +---rw algorithm
|        |             | isa:symmetric
| ric-algorithm-type
|        |     +---rw key-format?
|        |       | identityref
|        |       |   +---rw (key-type)
|        |       |     +---:(key)
|        |       |       +---rw key?
|        |       |         | bina\n| ry
|        |     +---:(hidden-key)
|        |       +---rw hidden\n| key?
|        |     empty
|        |   +---:(encrypted-
| key)
|        |     empty
|       +---rw encryp\n| ted-key
| y-type)
|        |     +---:(s\n| symmetric-key-ref)
|        |     +---:(a\n| rw symmetric-key-ref? leafref
|        |     +---\n| (keystore-supported)?
| symmetric-key-ref
|        |     +---\n| rw asymmetric-key-ref? leafref
|        |     +---\n| (keystore-supported)?
| ue?
| binary
|       +---:(keystore)
| rted)?
++-rw idle-timeout?  uint16
++-rw endpoint* [name]
  +++-rw name        string
  +++-rw (transport)
    +--:(ssh) {ssh-listen}?
      +++-rw ssh
        +++-rw tcp-server-parameters
          +++-rw local-address  inet:ip-address
          +++-rw local-port?    inet:port-number
          +++-rw keepalives! {keepalives-supported}?
            +++-rw idle-time   uint16
            +++-rw max-probes  uint16
            +++-rw probe-interval  uint16
        +++-rw ssh-client-parameters
          +++-rw client-identity
            +++-rw username?    string
            +++-rw (auth-type)
              +--:(password)
                |  +++-rw password?  string
              +--:(public-key)
                |  +++-rw public-key
                  |    +++-rw (local-or-keystore)
                  |      +--:(local)
                  |      |    {local-definitions-suported}?
        \pported)?
        \algorithm-type
          +++-rw public-key-format?
            |    identityref
          +++-rw public-key
            |    binary
          +++-rw private-key-format?
            |    identityref
          +++-rw (private-key-type)
            +--:(private-key)
              |  +++-rw private-key?
                |    binary
                |    :+::(hidden-private-key)
        \key)
        \te-key?
          |  +++-rw hidden-private
        \e-key)
          |  empty
        \ivate-key
          +++-rw encrypted-private
\iveate-key
  | | | | | +--rw encrypted-pr\n\lvicate-key
  | | | | +--rw (key-type)
\le-key-ref)
  | | | | +--:(symmetr\n\lmetric-key-ref? leafref
  | | | | \n\keystore-supported)?
  | | | | +--:(asymmet\n\lric-key-ref)
  | | | | +--rw asy\n\lmmetric-key-ref? leafref
  | | | | \n\keystore-supported)?
  | | | | +--rw value?
    | | | | +--rw cert?
    | | | | end-entity-cert-\n\lcms
  | | | +----n certificate-expira\n\ltion
  | | | +-- expiration-date
    | | | yang:date-and\n\lt-time
  | | | +----x generate-certifica\n\lte-signing-request
  | | | +----w input
    | | | +----w subject
    | | | binary
    | | | +----w attributes?
    | | | binary
    | | | +--ro output
    | | | +--ro certificate-\n\lsigning-request
  | | | binary
  | | +--:(keystore)
    | | {keystore-supported)?
    | | +--rw keystore-reference
    | | +--rw asymmetric-key?
    | | ks:asymmetric-ke\n\ly-ref
  | | | +--rw server-authentication
    | | +--rw ssh-host-keys!
    | | +--rw (local-or-truststore)
    | | +--:(local)
\d)?
  |     |  |  |     |  {local-definitions-supported}?
  |     |  |  |     |  +--rw local-definition
  |     |  |  |     |       +--rw host-key
  |     |  |  |     |         ct:ssh-host-key
  |     |  |  |     |       +--:(truststore)
  |     |  |  |     |           {truststore-supported, ssh-h}
\host-keys)?
  |     |  |  |     |       +--rw truststore-reference?
  |     |  |  |     |          ts:host-keys-ref
  |     |  |  |     |       +--rw ca-certs! {sshcmn:ssh-x509-certs}?
  |     |  |  |     |          +--rw (local-or-truststore)
  |     |  |  |     |              +--:(local)
  |     |  |  |     |        {local-definitions-supported}?
\d)?
  |     |  |  |     |       +--rw local-definition
  |     |  |  |     |          +--rw cert*
  |     |  |  |     |               trust-anchor-cert-cms
  |     |  |  |     |               +--n certificate-expiration
  |     |  |  |     |                   +-- expiration-date
  |     |  |  |     |                    yang:date-and-time
  |     |  |  |     |       +--:(truststore)
  |     |  |  |     |        {truststore-supported, x509-}
\certificates)?
  |     |  |  |     |       +--rw truststore-reference?
  |     |  |  |     |          ts:certificates-ref
  |     |  |  |     |       +--rw server-certs!
  |     |  |  |     |          {sshcmn:ssh-x509-certs}?
  |     |  |  |     |          +--rw (local-or-truststore)
  |     |  |  |     |              +--:(local)
  |     |  |  |     |        {local-definitions-supported}?
\d)?
  |     |  |  |     |       +--rw local-definition
  |     |  |  |     |          +--rw cert*
  |     |  |  |     |               trust-anchor-cert-cms
  |     |  |  |     |               +--n certificate-expiration
  |     |  |  |     |                   +-- expiration-date
  |     |  |  |     |                    yang:date-and-time
  |     |  |  |     |       +--:(truststore)
  |     |  |  |     |        {truststore-supported, x509-}
\certificates)?
  |     |  |  |     |       +--rw truststore-reference?
  |     |  |  |     |          ts:certificates-ref
  |     |  |  |     |       +--rw transport-params
  |     |  |  |     |          {ssh-client-transport-params-config}?
  |     |  |  |     |          +--rw host-key
  |     |  |  |     |              +--rw host-key-alg* identityref
  |     |  |  |     |              +--rw key-exchange
+++rw key-exchange-alg* identityref
  +++rw encryption
    +++rw encryption-alg* identityref
    +++rw mac
      +++rw mac-alg* identityref
    +++rw keepalives! (ssh-client-keepalives)?
    +++rw max-wait? uint16
    +++rw max-attempts? uint8
  +++rw netconf-client-parameters
  +++:(tls) (tls-listen)?
    +++rw tls
      +++rw tcp-server-parameters
        +++rw local-address inet:ip-address
        +++rw local-port? inet:port-number
        +++rw keepalives! (keepalives-supported)?
          +++rw idle-time uint16
          +++rw max-probes uint16
          +++rw probe-interval uint16
      +++rw tls-client-parameters
        +++rw client-identity
          +++rw (auth-type)
            +++:(certificate)
              +++rw certificate
                +++x509-certificate-auth)?
                  +++rw (local-or-keystore)
                    +++:(local)
                      {local-definitions-supported}?
                      +++rw local-definition
                        +++rw algorithm
                          iasa:asymmetric-
                  
algorithm-type
          +++rw public-key-format?
            identityref
          +++rw public-key
            binary
          +++rw private-key-format?
            identityref
          +++rw (private-key-type)
            +++:(private-key)
              +++rw private-key?
                binary
                +++:(hidden-private-key)
      
key
          +++rw hidden-private?
            empty
            +++:(encrypted-private-key)
\( \text{encrypted-pr} \)

\( \text{key-type} \)

\( \text{symmetric-key-ref} \)

\( \text{asymmetric-key-ref} \)

\( \text{keystore-supported} \)

\( \text{certificate-expiration} \)

\( \text{generate-certificates} \)

\( \text{keystore-supported} \)

\( \text{value} \)

\( \text{cert} \)

\( \text{end-entity-cert} \)

\( \text{expiration-date} \)

\( \text{generate-certificates} \)

\( \text{keystore} \)

\( \text{subject} \)

\( \text{attributes} \)

\( \text{output} \)

\( \text{certificate} \)

\( \text{keystore-reference} \)

\( \text{asymmetric-key} \)

\( \text{raw-public-key} \)

\( \text{raw-public-key-auth} \)
++rw psk (psk-auth)?
    ++rw (local-or-keystore)
      +++:(local)
        {local-definitions-supported}?

++rw local-definition
    ++rw algorithm
      isa:symmetric-algorithm-type
        ++rw key-format?
          identityref
        ++rw (key-type)
          +++:(key)
            ++rw key?
              binary
            +++:(hidden-key)
              +++rw hidden-key?
                empty
            +++:(encrypted-key)
              +++rw encrypted-key
                ++rw (key-type)
                  +++:(symmetric-key-ref)
A.2. Expanded Tree Diagram for ‘ietf-netconf-server’

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-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 4.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

========== NOTE: ‘\’ line wrapping per BCP XXX (RFC XXXX) =========
\metric-key-ref
\asymmetric-key-ref? leafref
\{keystore-supported)? ++rw value?
\ry
\rt-cms
\iration
\and-time
\icate-signing-request
\s?
\te-signing-request
\d)?\-key-ref
\ leafref
\ publickey? empty
\ password? empty
\ hostbased? empty
\ none? empty
\ other* string
\ users \{client-auth-config-supported)?
\ name string
\ password? ianach:cyphar-hash
\texttt{id\textasciitilde}\texttt{t\textasciitilde}ref
\texttt{\textasciitilde}rw public-key
\texttt{\textasciitilde}rw private-key-format?
\texttt{\textasciitilde}rw (private-key-type)
\texttt{\textasciitilde}rw private-key?
\texttt{\textasciitilde}rw hidden-private-key?
\texttt{\textasciitilde}rw encrypted-private-key
\texttt{\textasciitilde}rw (key-type)
\texttt{\textasciitilde}rw symmetric-key-ref?
\texttt{\textasciitilde}rw asymmetric-key-ref?
\texttt{\textasciitilde}rw value?
\texttt{\textasciitilde}rw cert?
\texttt{\textasciitilde}n certificate-expiration
\texttt{\textasciitilde}x generate-certificate
\texttt{\textasciitilde}w input
\texttt{\textasciitilde}w subject
\texttt{\textasciitilde}w attributes?
++-rw local-port? inet:port-number
  (local-binding-supported)?
++-rw keepalives!
  (keepalives-supported)?
    +-rw idle-time uint16
    +-rw max-probes uint16
    +-rw probe-interval uint16
++-rw ssh-server-parameters
  ++-rw server-identity
    +-rw host-key* [name]
      ++-rw name string
      ++-rw (host-key-type)
        ++-: (public-key)
          ++-rw public-key
          ++-rw (local-or-keystore)
            ++-: (local)
              (local-definitions-supported)?
      (host-key-type)
        ++-: (private-key)
        (private-key-type)
        ++-: (encrypt)

Watsen                      Expires May 23, 2020                      [Page 77]
Internet-Draft      NETCONF Client and Server Models       November 2019

\certificate-signing-request

\ject

\inary

\ributes?

\inary

\ttificate-signing-request

\inary

\pported)?

\erence

\c-key?

\metric-key-ref

\te?  leafref

\ds

\d)?

\ons-supported)?

Watsen                    Expires May 23, 2020                 [Page 80]
\binary
|              |  |              |                      b\n|              |  |              +--:(keystore)
|              |  |                       {keystore-supported}?\n|              |  |                 +--rw keystore-reference?\n|              |  |                         ks:symmetric-\n|              |  +--rw client-authentication!
|              |  |       {client-auth-config-supported}?\n|              |  |  +--rw ca-certs!
|              |  |                      {x509-certificate-auth}?\n|              |  |  +--rw (local-or-truststore)
|              |  |                         +--:(local)
|              |  |                       {local-definitions-supported}?\n|              |  |  +--rw local-definition
|              |  |  +--rw cert*
|              |  |       trust-anchor-certificate\n|              |  +--n certificate-expiration\n|              |  +-- expiration-date
|              |  |                yang:date-and-time\n|              |  +--:(truststore)
|              |  |              {truststore-supported}\n|              |  |        +--rw truststore-reference?\n|              |  |                       ts:certificates-ref\n|              |  |  +--rw client-certs!
|              |  |                      {x509-certificate-auth}?\n|              |  |  +--rw (local-or-truststore)
|              |  |                         +--:(local)
|              |  |                       {local-definitions-supported}?\n|              |  |  +--rw local-definition
|              |  |  +--rw cert*
|              |  |       trust-anchor-certificate\n|              |  +--n certificate-expiration\n|              |  +-- expiration-date
|              |  |                yang:date-and-time
|              |  +--:(truststore)
|              |  |              {truststore-supported}
Internet-Draft      NETCONF Client and Server Models       November 2019

\,x509-certificates}\n|              |  |  |        +--rw truststore-reference?
   ts:certificates-ref
   +--rw raw-public-keys!
      {raw-public-key-auth}\n      +--rw (local-or-truststore)
         +--:(local)
            {local-definitions-supported}\n\pported)?
|              |  |  |        +--rw local-definition
   +--rw raw-public-key*   [name]
      +--rw name
         string
      +--rw algorithm
         iasa:asymmetric\n\ic-algorithm-type
|              |  |  |        +--rw public-key-form\n\at?|              |  |  |        | identityref
   --rw public-key
      binary
   +--:(truststore)
      {truststore-supported}\n\,raw-public-keys}\n|              |  |  |        +--rw truststore-reference?
   ts:raw-public-keys-\n\ref |              |  |  |        +--rw hello-params
   (tls-server-hello-params-config\n\})?|              |  |  |        +--rw tls-versions
   +--rw tls-version* identityref
   +--rw cipher-suites
      +--rw cipher-suite* identityref
   +--rw keepalives!
      {tls-server-keepalives}? 
      +--rw max-wait? uint16
      +--rw max-attempts? uint8
   +--rw netconf-server-parameters
      +--rw client-identity-mappings
         {tls-listen or tls-call-home or\n\ sshcmn:ssh-x509-certs}\n|              |  |  |        +--rw cert-to-name* [id]
   +--rw id uint32
   +--rw fingerprint?
      x509c2ntls-fingerprint
   +--rw map-type identityref
Appendix B. Change Log

B.1. 00 to 01

- Renamed "keychain" to "keystore".

B.2. 01 to 02

- Added to ietf-netconf-client ability to connected to a cluster of endpoints, including a reconnect-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.

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 ‘netconf-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-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.

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" leaves.
- 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 "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.
B.13. 12 to 13

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

B.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)

B.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.

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.

B.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.

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, Alan Luchuk, Ladislav Lhotka, 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