

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 4, 2018

K. Watsen  
Juniper Networks  
G. Wu  
Cisco Networks  
J. Schoenwaelder  
Jacobs University Bremen  
July 3, 2017

**NETCONF Client and Server Models**  
**draft-ietf-netconf-netconf-client-server-04**

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:

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

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

- o "XXXX" --> the assigned RFC value for this draft
- o "YYYY" --> the assigned RFC value for I-D.ietf-netconf-ssh-client-server
- o "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:

- o "2017-07-03" --> the publication date of this draft

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

- o [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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on January 4, 2018.

#### Copyright Notice

Copyright (c) 2017 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 (<http://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

<a href="#">1.</a>	<a href="#">Introduction</a>	<a href="#">4</a>
<a href="#">1.1.</a>	<a href="#">Terminology</a>	<a href="#">4</a>
<a href="#">1.2.</a>	<a href="#">Tree Diagrams</a>	<a href="#">4</a>
<a href="#">2.</a>	<a href="#">The NETCONF Client Model</a>	<a href="#">4</a>
<a href="#">2.1.</a>	<a href="#">Tree Diagram</a>	<a href="#">5</a>
<a href="#">2.2.</a>	<a href="#">Example Usage</a>	<a href="#">8</a>
<a href="#">2.3.</a>	<a href="#">YANG Model</a>	<a href="#">10</a>
<a href="#">3.</a>	<a href="#">The NETCONF Server Model</a>	<a href="#">20</a>
<a href="#">3.1.</a>	<a href="#">Tree Diagram</a>	<a href="#">20</a>
<a href="#">3.2.</a>	<a href="#">Example Usage</a>	<a href="#">23</a>
<a href="#">3.3.</a>	<a href="#">YANG Model</a>	<a href="#">26</a>
<a href="#">4.</a>	<a href="#">Design Considerations</a>	<a href="#">37</a>
<a href="#">4.1.</a>	<a href="#">Support all NETCONF transports</a>	<a href="#">38</a>
<a href="#">4.2.</a>	<a href="#">Enable each transport to select which keys to use</a>	<a href="#">38</a>
<a href="#">4.3.</a>	<a href="#">Support authenticating NETCONF clients certificates</a>	<a href="#">38</a>
<a href="#">4.4.</a>	<a href="#">Support mapping authenticated NETCONF client certificates to usernames</a>	<a href="#">38</a>
<a href="#">4.5.</a>	<a href="#">Support both listening for connections and call home</a>	<a href="#">38</a>
<a href="#">4.6.</a>	<a href="#">For Call Home connections</a>	<a href="#">39</a>
<a href="#">4.6.1.</a>	<a href="#">Support more than one NETCONF client</a>	<a href="#">39</a>
<a href="#">4.6.2.</a>	<a href="#">Support NETCONF clients having more than one endpoint</a>	<a href="#">39</a>
<a href="#">4.6.3.</a>	<a href="#">Support a reconnection strategy</a>	<a href="#">39</a>
<a href="#">4.6.4.</a>	<a href="#">Support both persistent and periodic connections</a>	<a href="#">39</a>
<a href="#">4.6.5.</a>	<a href="#">Reconnection strategy for periodic connections</a>	<a href="#">40</a>
<a href="#">4.6.6.</a>	<a href="#">Keep-alives for persistent connections</a>	<a href="#">40</a>
<a href="#">4.6.7.</a>	<a href="#">Customizations for periodic connections</a>	<a href="#">40</a>
<a href="#">5.</a>	<a href="#">Security Considerations</a>	<a href="#">40</a>
<a href="#">6.</a>	<a href="#">IANA Considerations</a>	<a href="#">41</a>
<a href="#">6.1.</a>	<a href="#">The IETF XML Registry</a>	<a href="#">41</a>
<a href="#">6.2.</a>	<a href="#">The YANG Module Names Registry</a>	<a href="#">42</a>
<a href="#">7.</a>	<a href="#">Acknowledgements</a>	<a href="#">42</a>
<a href="#">8.</a>	<a href="#">References</a>	<a href="#">42</a>
<a href="#">8.1.</a>	<a href="#">Normative References</a>	<a href="#">42</a>
<a href="#">8.2.</a>	<a href="#">Informative References</a>	<a href="#">43</a>
<a href="#">Appendix A.</a>	<a href="#">Change Log</a>	<a href="#">45</a>
<a href="#">A.1.</a>	<a href="#">server-model-09 to 00</a>	<a href="#">45</a>
<a href="#">A.2.</a>	<a href="#">00 to 01</a>	<a href="#">45</a>
<a href="#">A.3.</a>	<a href="#">01 to 02</a>	<a href="#">45</a>
<a href="#">A.4.</a>	<a href="#">02 to 03</a>	<a href="#">45</a>
<a href="#">A.5.</a>	<a href="#">03 to 04</a>	<a href="#">45</a>
	<a href="#">Authors' Addresses</a>	<a href="#">46</a>



## **1. Introduction**

This document defines two YANG [\[RFC7950\]](#) 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.

NETCONF is defined by [\[RFC6241\]](#). SSH is defined by [\[RFC4252\]](#), [\[RFC4253\]](#), and [\[RFC4254\]](#). TLS is defined by [\[RFC5246\]](#). NETCONF Call Home is defined by [\[RFC8071\]](#).

### **1.1. Terminology**

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

### **1.2. Tree Diagrams**

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "\*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## **2. 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.



This model supports both the SSH and TLS transport protocols, using the SSH client and TLS client groupings defined in [\[I-D.ietf-netconf-ssh-client-server\]](#) and [\[I-D.ietf-netconf-tls-client-server\]](#) respectively.

All private keys and trusted certificates are held in the keystore model defined in [\[I-D.ietf-netconf-keystore\]](#).

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

## 2.1. Tree Diagram

Just the container is displayed below, but there is also a grouping that the container is using.

Note: all lines are folded at column 71 with no '\\' character.

```

module: ietf-netconf-client
  +--rw netconf-client
    +--rw initiate {initiate}?
      | +--rw netconf-server* [name]
      |   +--rw name string
      |   +--rw (transport)
      |     | +--:(ssh) {ssh-initiate}?
      |     |   +--rw ssh
      |     |     +--rw endpoints
      |     |       +--rw endpoint* [name]
      |     |         +--rw name string
      |     |         +--rw address inet:host
      |     |         +--rw port? inet:port-number
      |     |   +--rw server-auth
      |     |     +--rw trusted-ssh-host-keys?
      |     |       | -> /ks:keystore/trusted-host-keys/name
      |     |     +--rw trusted-ca-certs? leafref
      |     |       | {sshcom:ssh-x509-certs}?
      |     |     +--rw trusted-server-certs? leafref
      |     |       | {sshcom:ssh-x509-certs}?
      |     |   +--rw client-auth
      |     |     +--rw username? string
      |     |     +--rw (auth-type)
      |     |       +--:(certificate)
      |     |       | +--rw certificate? leafref
      |     |       |   {sshcom:ssh-x509-certs}?
      |     |       +--:(public-key)
      |     |       | +--rw public-key?
      |     |       |   -> /ks:keystore/keys/key/name
      |     |       +--:(password)

```





```

|         |         |         |         +--rw password?          string
|         |         |         | +--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 compression
|         |         |         |         | +--rw compression-alg*   identityref
|         |         | +---:(tls) {tls-initiate}?
|         |         |         +--rw tls
|         |         |         | +--rw endpoints
|         |         |         |         | +--rw endpoint* [name]
|         |         |         |         |         +--rw name          string
|         |         |         |         |         +--rw address       inet:host
|         |         |         |         |         +--rw port?         inet:port-number
|         |         |         |         +--rw server-auth
|         |         |         |         | +--rw trusted-ca-certs?     leafref
|         |         |         |         | +--rw trusted-server-certs? leafref
|         |         |         |         +--rw client-auth
|         |         |         |         | +--rw (auth-type)
|         |         |         |         |         +---:(certificate)
|         |         |         |         |         | +--rw certificate?  leafref
|         |         |         |         +--rw hello-params
|         |         |         |         |         {tls-client-hello-params-config}?
|         |         |         |         |         +--rw tls-versions
|         |         |         |         |         | +--rw tls-version*  identityref
|         |         |         |         |         +--rw cipher-suites
|         |         |         |         |         | +--rw cipher-suite*  identityref
|         |         | +--rw connection-type
|         |         |         +--rw (connection-type)?
|         |         |         | +---:(persistent-connection)
|         |         |         |         | +--rw persistent!
|         |         |         |         |         +--rw idle-timeout?   uint32
|         |         |         |         |         +--rw keep-alives
|         |         |         |         |         | +--rw max-wait?      uint16
|         |         |         |         |         | +--rw max-attempts?  uint8
|         |         |         |         +---:(periodic-connection)
|         |         |         |         | +--rw periodic!
|         |         |         |         |         +--rw idle-timeout?    uint16
|         |         |         |         |         +--rw reconnect-timeout? uint16
|         |         | +--rw reconnect-strategy
|         |         |         +--rw start-with?      enumeration
|         |         |         +--rw max-attempts?    uint8

```



```

+--rw listen {listen}?
  +--rw max-sessions?   uint16
  +--rw idle-timeout?   uint16
  +--rw endpoint* [name]
    +--rw name          string
    +--rw (transport)
      +--:(ssh) {ssh-listen}?
        | +--rw ssh
        |   +--rw address?          inet:ip-address
        |   +--rw port?             inet:port-number
        |   +--rw server-auth
        |   | +--rw trusted-ssh-host-keys?
        |   | | -> /ks:keystore/trusted-host-keys/name
        |   | +--rw trusted-ca-certs?   leafref
        |   | | {sshcom:ssh-x509-certs}?
        |   | +--rw trusted-server-certs? leafref
        |   | | {sshcom:ssh-x509-certs}?
        |   +--rw client-auth
        |   | +--rw username?         string
        |   | +--rw (auth-type)
        |   | | +--:(certificate)
        |   | | | +--rw certificate?   leafref
        |   | | | | {sshcom:ssh-x509-certs}?
        |   | | +--:(public-key)
        |   | | | +--rw public-key?
        |   | | | | -> /ks:keystore/keys/key/name
        |   | | +--:(password)
        |   | | +--rw password?        string
        |   +--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 compression
        |   | | +--rw compression-alg* identityref
        +--:(tls) {tls-listen}?
          +--rw tls
            +--rw address?          inet:ip-address
            +--rw port?             inet:port-number
            +--rw server-auth
            | +--rw trusted-ca-certs?   leafref
            | +--rw trusted-server-certs? leafref
            +--rw client-auth

```



```
|  +--rw (auth-type)
|    +--:(certificate)
|      +--rw certificate?  leafref
+--rw hello-params
      {tls-client-hello-params-config}?
+--rw tls-versions
|  +--rw tls-version*  identityref
+--rw cipher-suites
      +--rw cipher-suite*  identityref
```

## **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 listening for call-home connections, again using both the SSH and TLS transport protocols.

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



```
<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>
      <ssh>
        <endpoints>
          <endpoint>
            <name>corp-fw1.example.com</name>
            <address>corp-fw1.example.com</address>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <address>corp-fw2.example.com</address>
          </endpoint>
        </endpoints>
        <server-auth>
          <trusted-server-certs>deployment-specific-ca-certs</trusted-server-
certs>
        </server-auth>
        <client-auth>
          <username>foobar</username>
          <public-key>ex-rsa-key</public-key>
        </client-auth>
      </ssh>
    </netconf-server>
  </initiate>

  <!-- endpoints to listen for NETCONF Call Home connections on -->
  <listen>
    <endpoint>
      <name>Intranet-facing listener</name>
      <ssh>
        <address>11.22.33.44</address>
        <server-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-server-certs>explicitly-trusted-server-certs</trusted-
server-certs>
          <trusted-ssh-host-keys>explicitly-trusted-ssh-host-keys</trusted-ssh-
host-keys>
        </server-auth>
        <client-auth>
          <username>foobar</username>
          <public-key>ex-rsa-key</public-key>
        </client-auth>
      </ssh>
```



```
</endpoint>  
</listen>  
</netconf-client>
```

Watsen, et al.

Expires January 4, 2018

[Page 9]

### **2.3. YANG Model**

This YANG module imports YANG types from [[RFC6991](#)] and [[RFC7407](#)].

```
<CODE BEGINS> file "ietf-netconf-client@2017-07-03.yang"

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

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix "ncc";

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

  import ietf-ssh-client {
    prefix ss;
    revision-date 2017-06-13; // stable grouping definitions
    reference
      "RFC YYYY: SSH Client and Server Models";
  }

  import ietf-tls-client {
    prefix ts;
    revision-date 2017-06-13; // stable grouping definitions
    reference
      "RFC ZZZZ: TLS Client and Server Models";
  }

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

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

    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>

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

  description
```



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

Copyright (c) 2014 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

```
revision "2017-07-03" {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Client and Server Models";
}

// Features

feature initiate {
  description
    "The 'initiate' feature indicates that the NETCONF client
    supports initiating NETCONF connections to NETCONF servers
    using at least one transport (e.g., SSH, TLS, etc.).";
}

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 listen {
  description
    "The 'listen' feature indicates that the NETCONF client
    supports opening a port to accept NETCONF server call
    home connections using at least one transport (e.g.,
    SSH, TLS, etc.).";
}

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";
}

container netconf-client {
  uses netconf-client;
  description
    "Top-level container for NETCONF client configuration.";
}

grouping netconf-client {
  description
    "Top-level grouping for NETCONF client configuration.";

  container initiate {
    if-feature initiate;
    description
      "Configures client initiating underlying TCP connections.";
    list netconf-server {
      key name;
      description
        "List of NETCONF servers the NETCONF client is to initiate
        connections to.";
      leaf name {
```



```
    type string;
    description
      "An arbitrary name for the NETCONF server.";
  }
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";

    case ssh {
      if-feature ssh-initiate;
      container ssh {
        description
          "Specifies SSH-specific transport configuration.";
        uses endpoints-container {
          refine endpoints/endpoint/port {
            default 830;
          }
        }
        uses ss:ssh-client-grouping;
      }
    } // end ssh

    case tls {
      if-feature tls-initiate;
      container tls {
        description
          "Specifies TLS-specific transport configuration.";
        uses endpoints-container {
          refine endpoints/endpoint/port {
            default 6513;
          }
        }
        uses ts:tls-client-grouping {
          refine "client-auth/auth-type" {
            mandatory true;
            description
              "NETCONF/TLS clients MUST pass some authentication
              credentials.";
          }
        }
      }
    } // end tls
  } // end transport

  container connection-type {
    description
```





```
"Indicates the kind of connection to use.";
choice connection-type {
  description
    "Selects between available connection types.";
  case persistent-connection {
    container persistent {
      presence true;
      description
        "Maintain a persistent connection to the NETCONF
        server. If the connection goes down, immediately
        start trying to reconnect to it, 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.";
    }
    leaf idle-timeout {
      type uint32;
      units "seconds";
      default 86400; // one day;
      description
        "Specifies the maximum number of seconds that a
        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 client will never drop
        a session because it is idle. Sessions that
        have a notification subscription active are
        never dropped.";
    }
  }
  container keep-alives {
    description
      "Configures the keep-alive policy, to proactively
      test the aliveness of the SSH/TLS server. An
      unresponsive SSH/TLS server will be dropped after
      approximately max-attempts * max-wait seconds.";
    reference
      "RFC 8071: NETCONF Call Home and RESTCONF Call
      Home, Section 3.1, item S6";
    leaf max-wait {
      type uint16 {
        range "1..max";
      }
      units seconds;
      default 30;
      description
        "Sets the amount of time in seconds after which
        if no data has been received from the SSH/TLS
```



```
        server, a SSH/TLS-level message will be sent
        to test the aliveness of the SSH/TLS server.";
    }
    leaf max-attempts {
        type uint8;
        default 3;
        description
            "Sets the maximum number of sequential keep-alive
            messages that can fail to obtain a response from
            the SSH/TLS server before assuming the SSH/TLS
            server is no longer alive.";
    }
}
}
}
case periodic-connection {
    container periodic {
        presence true;
        description
            "Periodically connect to the NETCONF server, so that
            the NETCONF server may deliver messages pending for
            the NETCONF client. The NETCONF server must close
            the connection when it is ready to release it. Once
            the connection has been closed, the NETCONF client
            will restart its timer until the next connection.";
        leaf idle-timeout {
            type uint16;
            units "seconds";
            default 300; // five minutes
            description
                "Specifies the maximum number of seconds that a
                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.";
        }
        leaf reconnect-timeout {
            type uint16 {
                range "1..max";
            }
            units minutes;
            default 60;
            description
                "Sets the maximum amount of unconnected time the
                NETCONF client will wait before re-establishing
```



```
        a connection to the NETCONF server. The NETCONF
        client may initiate a connection before this
        time if desired (e.g., to set configuration).";
    }
  }
}
}
}
}
}
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.";
      }
    }
  }
  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).";
    }
}
} // end netconf-server
} // end initiate

container listen {
  if-feature listen;
  description
    "Configures client accepting call-home TCP connections.";

  leaf max-sessions {
    type uint16;
    default 0;
    description
      "Specifies the maximum number of concurrent sessions
       that can be active at one time. The value 0 indicates
       that no artificial session limit should be used.";
  }

  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;
    description
      "List of endpoints to listen for NETCONF connections.";
    leaf name {
      type string;
      description
        "An arbitrary name for the NETCONF listen endpoint.";
    }
    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.";
      leaf address {
        type inet:ip-address;
        description
          "The IP address to listen for call-home connections.";
      }
      leaf port {
        type inet:port-number;
        default 4334;
        description
          "The port number to listen for call-home connections.";
      }
      uses ss:ssh-client-grouping;
    }
  }
  case tls {
    if-feature tls-listen;
    container tls {
      description
        "TLS-specific listening configuration for inbound
        connections.";
      leaf address {
        type inet:ip-address;
        description
          "The IP address to listen for call-home connections.";
      }
      leaf port {
        type inet:port-number;
        default 4335;
        description
          "The port number to listen for call-home connections.";
      }
      uses ts:tls-client-grouping {
        refine "client-auth/auth-type" {
          mandatory true;
          description
            "NETCONF/TLS clients MUST pass some authentication
            credentials.";
        }
      }
    }
  }
} // end transport
} // end endpoint
} // end listen
```



```
} // end netconf-client

grouping endpoints-container {
  description
    "This grouping is used to configure a set of NETCONF servers
    a NETCONF client may initiate connections to.";
  container endpoints {
    description
      "Container for the list of endpoints.";
    list endpoint {
      key name;
      unique "address port";
      min-elements 1;
      ordered-by user;
      description
        "A non-empty user-ordered list of endpoints for this NETCONF
        client to try to connect to. Defining more than one enables
        high-availability.";
      leaf name {
        type string;
        description
          "An arbitrary name for this endpoint.";
      }
      leaf address {
        type inet:host;
        mandatory true;
        description
          "The IP address or hostname of the endpoint. If a
          hostname is configured and the DNS resolution results
          in more than one IP address, the NETCONF client
          will process the IP addresses as if they had been
          explicitly configured in place of the hostname.";
      }
      leaf port {
        type inet:port-number;
        description
          "The IP port for this endpoint. The NETCONF client will
          use the IANA-assigned well-known port (set via a refine
          statement when uses) if no value is specified.";
      }
    }
  }
}
```

<CODE ENDS>



### 3. The NETCONF Server Model

The NETCONF server model presented in this section supports servers both listening for connections as well as initiating call-home connections.

This model supports both the SSH and TLS transport protocols, using the SSH server and TLS server groupings defined in [\[I-D.ietf-netconf-ssh-client-server\]](#) and [\[I-D.ietf-netconf-tls-client-server\]](#) respectively.

All private keys and trusted certificates are held in the keystore model defined in [\[I-D.ietf-netconf-keystore\]](#).

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

#### 3.1. Tree Diagram

Just the container is displayed below, but there is also a grouping that the container is using.

Note: all lines are folded at column 71 with no '\\' character.

```
module: ietf-netconf-server
  +--rw netconf-server
    +--rw session-options
      | +--rw hello-timeout?  uint16
    +--rw listen {listen}?
      | +--rw max-sessions?   uint16
      | +--rw idle-timeout?   uint16
      | +--rw endpoint* [name]
      |   +--rw name          string
      |   +--rw (transport)
      |     +--:(ssh) {ssh-listen}?
      |       | +--rw ssh
      |         | +--rw address?          inet:ip-address
      |         | +--rw port?             inet:port-number
      |         | +--rw host-keys
      |         |   +--rw host-key* [name]
      |         |     +--rw name          string
      |         |     +--rw (host-key-type)
      |         |       +--:(public-key)
      |         |         | +--rw public-key?
      |         |         |   -> /ks:keystore/keys/key/name
      |         |         +--:(certificate)
      |         |           +--rw certificate?  leafref
      |         |             {sshcom:ssh-x509-certs}?
      |         |
```



```

|         |         +--rw client-cert-auth {sshcom:ssh-x509-certs}?
|         |         |   +--rw trusted-ca-certs?      leafref
|         |         |   +--rw trusted-client-certs?   leafref
|         |         +--rw transport-params
|         |         |   {ssh-server-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 compression
|         |         |   |   +--rw compression-alg* identityref
|         |         +---:(tls) {tls-listen}?
|         |         +--rw tls
|         |         |   +--rw address?              inet:ip-address
|         |         |   +--rw port?                inet:port-number
|         |         |   +--rw certificates
|         |         |   |   +--rw certificate* [name]
|         |         |   |   |   +--rw name        leafref
|         |         |   +--rw client-auth
|         |         |   |   +--rw trusted-ca-certs?      leafref
|         |         |   |   +--rw trusted-client-certs?   leafref
|         |         |   |   +--rw cert-maps
|         |         |   |   |   +--rw cert-to-name* [id]
|         |         |   |   |   |   +--rw id          uint32
|         |         |   |   |   |   +--rw fingerprint x509c2n:tls-fingerprint
|         |         |   |   |   |   +--rw map-type     identityref
|         |         |   |   |   |   +--rw name        string
|         |         |   +--rw hello-params
|         |         |   |   {tls-server-hello-params-config}?
|         |         |   |   +--rw tls-versions
|         |         |   |   |   +--rw tls-version*    identityref
|         |         |   |   +--rw cipher-suites
|         |         |   |   |   +--rw cipher-suite*   identityref
+--rw call-home {call-home}?
+--rw netconf-client* [name]
+--rw name                string
+--rw (transport)
|   +---:(ssh) {ssh-call-home}?
|   |   +--rw ssh
|   |   |   +--rw endpoints
|   |   |   |   +--rw endpoint* [name]
|   |   |   |   |   +--rw name        string
|   |   |   |   |   +--rw address     inet:host
|   |   |   |   |   +--rw port?      inet:port-number

```





```

| | +--rw host-keys
| | | +--rw host-key* [name]
| | | | +--rw name string
| | | | +--rw (host-key-type)
| | | | | +--:(public-key)
| | | | | | +--rw public-key?
| | | | | | -> /ks:keystore/keys/key/name
| | | | | +--:(certificate)
| | | | | +--rw certificate? leafref
| | | | | {sshcom:ssh-x509-certs}?
| | +--rw client-cert-auth {sshcom:ssh-x509-certs}?
| | | +--rw trusted-ca-certs? leafref
| | | +--rw trusted-client-certs? leafref
| | +--rw transport-params
| | | {ssh-server-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 compression
| | | | +--rw compression-alg* identityref
| | +--:(tls) {tls-call-home}?
| | +--rw tls
| | | +--rw endpoints
| | | | +--rw endpoint* [name]
| | | | | +--rw name string
| | | | | +--rw address inet:host
| | | | | +--rw port? inet:port-number
| | | +--rw certificates
| | | | +--rw certificate* [name]
| | | | | +--rw name leafref
| | | +--rw client-auth
| | | | +--rw trusted-ca-certs? leafref
| | | | +--rw trusted-client-certs? leafref
| | | | +--rw cert-maps
| | | | | +--rw cert-to-name* [id]
| | | | | | +--rw id uint32
| | | | | | +--rw fingerprint x509c2n:tls-fingerprint
| | | | | | +--rw map-type identityref
| | | | | | +--rw name string
| | | +--rw hello-params
| | | | {tls-server-hello-params-config}?
| | | +--rw tls-versions
| | | | +--rw tls-version* identityref

```



```

|           +--rw cipher-suites
|           +--rw cipher-suite*  identityref
+--rw connection-type
|   +--rw (connection-type)?
|   +--:(persistent-connection)
|   |   +--rw persistent!
|   |   +--rw idle-timeout?    uint32
|   |   +--rw keep-alives
|   |   +--rw max-wait?        uint16
|   |   +--rw max-attempts?    uint8
|   +--:(periodic-connection)
|   +--rw periodic!
|   +--rw idle-timeout?        uint16
|   +--rw reconnect-timeout?    uint16
+--rw reconnect-strategy
    +--rw start-with?    enumeration
    +--rw max-attempts?  uint8

```

### 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-keystore](#)].

```

<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">

  <!-- listening for SSH and TLS connections -->
  <listen>
    <endpoint> <!-- listening for SSH connections -->
      <name>netconf/ssh</name>
      <ssh>
        <address>11.22.33.44</address>
        <host-keys>
          <host-key>
            <name>public-key</name>
            <public-key>ex-rsa-key</public-key>
          </host-key>
          <host-key>
            <name>certificate</name>
            <certificate>builtin-idevid-cert</certificate>
          </host-key>
        </host-keys>

```



```
<client-cert-auth>
  <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
  <trusted-client-certs>explicitly-trusted-client-certs</trusted-
client-certs>
</client-cert-auth>
</ssh>
</endpoint>
<endpoint> <!-- listening for TLS sessions -->
  <name>netconf/tls</name>
  <tls>
    <address>11.22.33.44</address>
    <certificates>
      <certificate>
        <name>tls-ec-cert</name>
      </certificate>
    </certificates>
    <client-auth>
      <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
      <trusted-client-certs>explicitly-trusted-client-certs</trusted-
client-certs>
      <cert-maps>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <fingerprint>B3:4F:A1:8C:54</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>scooby-doo</name>
        </cert-to-name>
      </cert-maps>
    </client-auth>
  </tls>
</endpoint>
</listen>

<!-- calling home to an SSH and TLS based NETCONF clients -->
<call-home>
  <netconf-client> <!-- SSH-based client -->
    <name>config-mgr</name>
    <ssh>
      <endpoints>
        <endpoint>
          <name>east-data-center</name>
          <address>11.22.33.44</address>
        </endpoint>
```

```
<endpoint>  
  <name>west-data-center</name>
```

Watsen, et al.

Expires January 4, 2018

[Page 24]

```
        <address>55.66.77.88</address>
      </endpoint>
    </endpoints>
    <host-keys>
      <host-key>
        <name>certificate</name>
        <certificate>builtin-idevid-cert</certificate>
      </host-key>
    </host-keys>
    <client-cert-auth>
      <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
      <trusted-client-certs>explicitly-trusted-client-certs</trusted-
client-certs>
    </client-cert-auth>
  </ssh>
  <connection-type>
    <periodic>
      <idle-timeout>300</idle-timeout>
      <reconnect-timeout>60</reconnect-timeout>
    </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>event-correlator</name>
  <tls>
    <endpoints>
      <endpoint>
        <name>east-data-center</name>
        <address>22.33.44.55</address>
      </endpoint>
      <endpoint>
        <name>west-data-center</name>
        <address>33.44.55.66</address>
      </endpoint>
    </endpoints>
    <certificates>
      <certificate>
        <name>tls-ec-cert</name>
      </certificate>
    </certificates>
    <client-auth>
      <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
      <trusted-client-certs>explicitly-trusted-client-certs</trusted-
client-certs>
```



```
<cert-maps>  
  <cert-to-name>
```

Watsen, et al.

Expires January 4, 2018

[Page 25]

```
        <id>1</id>
        <fingerprint>11:0A:05:11:00</fingerprint>
        <map-type>x509c2n:san-any</map-type>
    </cert-to-name>
    <cert-to-name>
        <id>2</id>
        <fingerprint>B3:4F:A1:8C:54</fingerprint>
        <map-type>x509c2n:specified</map-type>
        <name>scooby-doo</name>
    </cert-to-name>
</cert-maps>
</client-auth>
</tls>
<connection-type>
    <persistent>
        <idle-timeout>300</idle-timeout>
        <keep-alives>
            <max-wait>30</max-wait>
            <max-attempts>3</max-attempts>
        </keep-alives>
    </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 Model

This YANG module imports YANG types from [\[RFC6991\]](#) and [\[RFC7407\]](#).

```
<CODE BEGINS> file "ietf-netconf-server@2017-07-03.yang"
```

```
module ietf-netconf-server {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix "ncs";

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



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

import ietf-ssh-server {
  prefix ss;
  revision-date 2017-06-13; // stable grouping definitions
  reference
    "RFC YYYY: SSH Client and Server Models";
}

import ietf-tls-server {
  prefix ts;
  revision-date 2017-06-13; // stable grouping definitions
  reference
    "RFC ZZZZ: TLS Client and Server Models";
}

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

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

  Author:   Kent Watsen
            <mailto:kwatsen@juniper.net>;

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

  Copyright (c) 2014 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";
```



```
revision "2017-07-03" {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Client and Server Models";
}

// Features

feature listen {
  description
    "The 'listen' feature indicates that the NETCONF server
    supports opening a port to accept NETCONF client connections
    using at least one transport (e.g., SSH, TLS, etc.).";
}

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 call-home {
  description
    "The 'call-home' feature indicates that the NETCONF server
    supports initiating NETCONF call home connections to NETCONF
    clients using at least one transport (e.g., SSH, TLS, etc.).";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

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";
}

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

grouping netconf-server {
  description
    "Top-level grouping for NETCONF server configuration.";

  container session-options { // SHOULD WE REMOVE THIS ALTOGETHER?
    description
      "NETCONF session options, independent of transport
      or connection strategy.";
    leaf hello-timeout {
      type uint16;
      units "seconds";
      default 600;
      description
        "Specifies the maximum number of seconds that a SSH/TLS
        connection may wait for a hello message to be received.
        A connection will be dropped if no hello message is
        received before this number of seconds elapses. If set
        to zero, then the server will wait forever for a hello
        message.";
    }
  }
}

container listen {
  if-feature listen;
  description
    "Configures listen behavior";
```





```
leaf max-sessions {
  type uint16;
  default 0;
  description
    "Specifies the maximum number of concurrent sessions
     that can be active at one time. The value 0 indicates
     that no artificial session limit should be used.";
}
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;
  description
    "List of endpoints to listen for NETCONF connections.";
  leaf name {
    type string;
    description
      "An arbitrary name for the NETCONF listen endpoint.";
  }
}

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.";
      leaf address {
        type inet:ip-address;
        description
          "The IP address of the interface to listen on. The
           SSH server will listen on all interfaces if no value
           is specified. Please note that some addresses have
           special meanings (e.g., '0.0.0.0' and '::').";
      }
    }
  }
}
```



```
    }
    leaf port {
      type inet:port-number;
      default 830;
      description
        "The local port number on this interface the SSH server
        listens on.";
    }
    uses ss:ssh-server-grouping;
  }
}
case tls {
  if-feature tls-listen;
  container tls {
    description
      "TLS-specific listening configuration for inbound
      connections.";
    leaf address {
      type inet:ip-address;
      description
        "The IP address of the interface to listen on. The
        TLS server will listen on all interfaces if no value
        is specified. Please note that some addresses have
        special meanings (e.g., '0.0.0.0' and '::').";
    }
    leaf port {
      type inet:port-number;
      default 6513;
      description
        "The local port number on this interface the TLS server
        listens on.";
    }
    uses ts:tls-server-grouping {
      refine "client-auth" {
        must 'trusted-ca-certs or trusted-client-certs';
        description
          "NETCONF/TLS servers MUST validate client
          certificates.";
      }
      augment "client-auth" {
        description
          "Augments in the cert-to-name structure.";
        uses cert-maps-grouping;
      }
    }
  }
}
}
```



```
    }
  }

  container call-home {
    if-feature call-home;
    description
      "Configures call-home behavior";
    list netconf-client {
      key name;
      description
        "List of NETCONF clients the NETCONF server is to initiate
        call-home connections to.";
      leaf name {
        type string;
        description
          "An arbitrary name for the remote NETCONF client.";
      }
    }
    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.";
          uses endpoints-container {
            refine endpoints/endpoint/port {
              default 4334;
            }
          }
          uses ss:ssh-server-grouping;
        }
      }
      case tls {
        if-feature tls-call-home;
        container tls {
          description
            "Specifies TLS-specific call-home transport
            configuration.";
          uses endpoints-container {
            refine endpoints/endpoint/port {
              default 4335;
            }
          }
          uses ts:tls-server-grouping {
            refine "client-auth" {
```



```

must trusted-ca-certs or trusted-client-certs';
description
    "NETCONF/TLS servers MUST validate client
    certificates.";
}
augment "client-auth" {
    description
        "Augments in the cert-to-name structure.";
    uses cert-maps-grouping;
}
}
}
}
}
container connection-type {
    description
        "Indicates the kind of connection to use.";
    choice connection-type {
        description
            "Selects between available connection types.";
        case persistent-connection {
            container persistent {
                presence true;
                description
                    "Maintain a persistent connection to the NETCONF
                    client. If the connection goes down, immediately
                    start trying to reconnect to it, 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.";
            }
            leaf idle-timeout {
                type uint32;
                units "seconds";
                default 86400; // one day;
                description
                    "Specifies the maximum number of seconds that a
                    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.";
            }
        }
        container keep-alives {
            description

```





```
    "Configures the keep-alive policy, to proactively
    test the aliveness of the SSH/TLS client.  An
    unresponsive SSH/TLS client will be dropped after
    approximately max-attempts * max-wait seconds.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call
    Home, Section 3.1, item S6";
  leaf max-wait {
    type uint16 {
      range "1..max";
    }
    units seconds;
    default 30;
    description
      "Sets the amount of time in seconds after which
      if no data has been received from the SSH/TLS
      client, a SSH/TLS-level message will be sent
      to test the aliveness of the SSH/TLS client.";
  }
  leaf max-attempts {
    type uint8;
    default 3;
    description
      "Sets the maximum number of sequential keep-alive
      messages that can fail to obtain a response from
      the SSH/TLS client before assuming the SSH/TLS
      client is no longer alive.";
  }
}
}
}
}
case periodic-connection {
  container periodic {
    presence true;
    description
      "Periodically connect to the NETCONF client, so that
      the NETCONF client may deliver messages pending for
      the NETCONF server.  The NETCONF client must close
      the connection when it is ready to release it.  Once
      the connection has been closed, the NETCONF server
      will restart its timer until the next connection.";
  }
  leaf idle-timeout {
    type uint16;
    units "seconds";
    default 300; // five minutes
    description
      "Specifies the maximum number of seconds that a
      a NETCONF session may remain idle.  A NETCONF
```



[illegible]



```
        endpoint connected to across reboots.";
    }
}
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).";
}
}
}
}
```

```
grouping cert-maps-grouping {
    description
        "A grouping that defines a container around the
        cert-to-name structure defined in RFC 7407.";
    container cert-maps {
        uses x509c2n:cert-to-name;
        description
            "The cert-maps container is used by a TLS-based NETCONF
            server to map the NETCONF client's presented X.509
            certificate to 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 WWW: NETCONF over TLS, Section 7";
    }
}
```

```
grouping endpoints-container {
    description
        "This grouping is used to configure a set of NETCONF clients
        a NETCONF server may initiate call-home connections to.";
```



```
container endpoints {
  description
    "Container for the list of endpoints.";
  list endpoint {
    key name;
    unique "address port";
    min-elements 1;
    ordered-by user;
    description
      "A non-empty user-ordered list of endpoints for this NETCONF
      server to try to connect to. Defining more than one enables
      high-availability.";
    leaf name {
      type string;
      description
        "An arbitrary name for this endpoint.";
    }
    leaf address {
      type inet:host;
      mandatory true;
      description
        "The IP address or hostname of the endpoint. If a
        hostname is configured and the DNS resolution results
        in more than one IP address, the NETCONF server
        will process the IP addresses as if they had been
        explicitly configured in place of the hostname.";
    }
    leaf port {
      type inet:port-number;
      description
        "The IP port for this endpoint. The NETCONF server will
        use the IANA-assigned well-known port (set via a refine
        statement when uses) if no value is specified.";
    }
  }
}
```

<CODE ENDS>

#### **4. Design Considerations**

Editorial: this section is a hold over from before, previously called "Objectives". It was only written to support the "server" (not the





"client"). The question is if it's better to add the missing "client" parts, or remove this section altogether.

The primary purpose of the YANG modules defined herein is to enable the configuration of the NETCONF client and servers. This scope includes the following objectives:

#### **4.1. Support all NETCONF transports**

The YANG module should support all current NETCONF transports, namely NETCONF over SSH [[RFC6242](#)], NETCONF over TLS [[RFC7589](#)], and to be extensible to support future transports as necessary.

Because implementations may not support all transports, the modules should use YANG "feature" statements so that implementations can accurately advertise which transports are supported.

#### **4.2. Enable each transport to select which keys to use**

Servers may have a multiplicity of host-keys or server-certificates from which subsets may be selected for specific uses. For instance, a NETCONF server may want to use one set of SSH host-keys when listening on port 830, and a different set of SSH host-keys when calling home. The data models provided herein should enable configuration of which keys to use on a per-use basis.

#### **4.3. Support authenticating NETCONF clients certificates**

When a certificate is used to authenticate a NETCONF client, there is a need to configure the server to know how to authenticate the certificates. The server should be able to authenticate the client's certificate either by using path-validation to a configured trust anchor or by matching the client-certificate to one previously configured.

#### **4.4. Support mapping authenticated NETCONF client certificates to usernames**

When a client certificate is used for TLS client authentication, the NETCONF server must be able to derive a username from the authenticated certificate. Thus the modules defined herein should enable this mapping to be configured.

#### **4.5. Support both listening for connections and call home**

The NETCONF protocols were originally defined as having the server opening a port to listen for client connections. More recently the NETCONF working group defined support for call-home ([[RFC8071](#)]),



enabling the server to initiate the connection to the client. Thus the modules defined herein should enable configuration for both listening for connections and calling home. Because implementations may not support both listening for connections and calling home, YANG "feature" statements should be used so that implementation can accurately advertise the connection types it supports.

#### **4.6. For Call Home connections**

The following objectives only pertain to call home connections.

##### **4.6.1. Support more than one NETCONF client**

A NETCONF server may be managed by more than one NETCONF client. For instance, a deployment may have one client for provisioning and another for fault monitoring. Therefore, when it is desired for a server to initiate call home connections, it should be able to do so to more than one client.

##### **4.6.2. Support NETCONF clients having more than one endpoint**

A NETCONF client managing a NETCONF server may implement a high-availability strategy employing a multiplicity of active and/or passive endpoint. Therefore, when it is desired for a server to initiate call home connections, it should be able to connect to any of the client's endpoints.

##### **4.6.3. Support a reconnection strategy**

Assuming a NETCONF client has more than one endpoint, then it becomes necessary to configure how a NETCONF server should reconnect to the client should it lose its connection to one the client's endpoints. For instance, the NETCONF server may start with first endpoint defined in a user-ordered list of endpoints or with the last endpoints it was connected to.

##### **4.6.4. Support both persistent and periodic connections**

NETCONF clients may vary greatly on how frequently they need to interact with a NETCONF server, how responsive interactions need to be, and how many simultaneous connections they can support. Some clients may need a persistent connection to servers to optimize real-time interactions, while others prefer periodic interactions in order to minimize resource requirements. Therefore, when it is necessary for server to initiate connections, it should be configurable if the connection is persistent or periodic.



#### **4.6.5. Reconnection strategy for periodic connections**

The reconnection strategy should apply to both persistent and periodic connections. How it applies to periodic connections becomes clear when considering that a periodic "connection" is a logical connection to a single server. That is, the periods of unconnectedness are intentional as opposed to due to external reasons. A periodic "connection" should always reconnect to the same server until it is no longer able to, at which time the reconnection strategy guides how to connect to another server.

#### **4.6.6. Keep-alives for persistent connections**

If a persistent connection is desired, it is the responsibility of the connection initiator to actively test the "aliveness" of the connection. The connection initiator must immediately work to reestablish a persistent connection as soon as the connection is lost. How often the connection should be tested is driven by NETCONF client requirements, and therefore keep-alive settings should be configurable on a per-client basis.

#### **4.6.7. Customizations for periodic connections**

If a periodic connection is desired, it is necessary for the NETCONF server to know how often it should connect. This frequency determines the maximum amount of time a NETCONF client may have to wait to send data to a server. A server may connect to a client before this interval expires if desired (e.g., to send data to a client).

### **5. Security Considerations**

A denial of service (DoS) attack MAY occur if the NETCONF server limits the maximum number of NETCONF sessions it will accept (i.e. the 'max-sessions' field in the `ietf-netconf-server` module is not zero) and either the "hello-timeout" or "idle-timeout" fields in `ietf-netconf-server` module have been set to indicate the NETCONF server should wait forever (i.e. set to zero).

The YANG module defined in this document uses groupings defined in [[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 module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [[RFC6241](#)] and RESTCONF [[RFC8040](#)]. Both of these protocols have mandatory-to-



implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [[RFC6536](#)] 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 this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

NONE

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

NONE

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

NONE

## **6. IANA Considerations**

### **[6.1.](#) The IETF XML Registry**

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

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.





## **6.2. The YANG Module Names Registry**

This document registers two YANG modules in the YANG Module Names registry [[RFC7950](#)]. Following the format in [[RFC7950](#)], the the following registrations are requested:

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

## **7. Acknowledgements**

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Phil Shafer, Sean Turner, and Bert Wijnen.

Juergen Schoenwaelder and was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

## **8. References**

### **8.1. Normative References**

- [I-D.ietf-netconf-keystore]  
Watsen, K., "Keystore Model", [draft-ietf-netconf-keystore-02](#) (work in progress), June 2017.
- [I-D.ietf-netconf-ssh-client-server]  
Watsen, K. and G. Wu, "SSH Client and Server Models", [draft-ietf-netconf-ssh-client-server-03](#) (work in progress), June 2017.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K. and G. Wu, "TLS Client and Server Models", [draft-ietf-netconf-tls-client-server-03](#) (work in progress), June 2017.



- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<http://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, <<http://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", [RFC 6991](#), DOI 10.17487/RFC6991, July 2013, <<http://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, <<http://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, <<http://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, <<http://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, <<http://www.rfc-editor.org/info/rfc8174>>.

## 8.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", [RFC 4252](#), DOI 10.17487/RFC4252, January 2006, <<http://www.rfc-editor.org/info/rfc4252>>.



- [RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", [RFC 4253](#), DOI 10.17487/RFC4253, January 2006, <<http://www.rfc-editor.org/info/rfc4253>>.
- [RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", [RFC 4254](#), DOI 10.17487/RFC4254, January 2006, <<http://www.rfc-editor.org/info/rfc4254>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [RFC 8040](#), DOI 10.17487/RFC8040, January 2017, <<http://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", [RFC 8071](#), DOI 10.17487/RFC8071, February 2017, <<http://www.rfc-editor.org/info/rfc8071>>.



## **Appendix A. Change Log**

### **A.1. server-model-09 to 00**

- o This draft was split out from [draft-ietf-netconf-server-model-09](#).
- o Added in previously missing ietf-netconf-client module.
- o Added in new features 'listen' and 'call-home' so future transports can be augmented in.

### **A.2. 00 to 01**

- o Renamed "keychain" to "keystore".

### **A.3. 01 to 02**

- o Added to ietf-netconf-client ability to connected to a cluster of endpoints, including a reconnection-strategy.
- o Added to ietf-netconf-client the ability to configure connection-type and also keep-alive strategy.
- o Updated both modules to accomodate new groupings in the ssh/tls drafts.

### **A.4. 02 to 03**

- o Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.
- o Changed 'netconf-client' to be a grouping (not a container).

### **A.5. 03 to 04**

- o Added [RFC 8174](#) to Requirements Language Section.
- o Replaced refine statement in ietf-netconf-client to add a mandatory true.
- o Added refine statement in ietf-netconf-server to add a must statement.
- o Now there are containers and groupings, for both the client and server models.





Authors' Addresses

Kent Watsen  
Juniper Networks

EMail: kwatsen@juniper.net

Gary Wu  
Cisco Networks

EMail: garywu@cisco.com

Juergen Schoenwaelder  
Jacobs University Bremen

EMail: j.schoenwaelder@jacobs-university.de

