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-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
- "YYYY" --> the assigned RFC value for I-D.ietf-netconf-ssh-client-server
- "ZZZZ" --> the assigned RFC value for I-D.ietf-netconf-tls-client-server
- "AAAA" --> the assigned RFC value for I-D.ietf-netconf-tcp-client-server
Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

- "2019-03-09" --> the publication date of this draft

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

- Appendix A. Change Log

Status of This Memo

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

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

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

This Internet-Draft will expire on September 10, 2019.

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.
Table of Contents

1. Introduction .................................................. 4
2. Terminology .................................................. 4
3. The NETCONF Client Model .................................... 4
   3.1. Tree Diagram ............................................ 4
   3.2. Example Usage ........................................... 13
   3.3. YANG Module ............................................. 16
4. The NETCONF Server Model .................................... 25
   4.1. Tree Diagram ............................................ 25
   4.2. Example Usage ........................................... 34
   4.3. YANG Module ............................................. 39
5. Design Considerations ........................................ 50
   5.1. Support all NETCONF transports ......................... 50
   5.2. Enable each transport to select which keys to use ...... 50
   5.3. Support authenticating NETCONF clients certificates ... 50
   5.4. Support mapping authenticated NETCONF client certificates to usernames ........................................... 51
   5.5. Support both listening for connections and call home ... 51
   5.6. For Call Home connections ................................ 51
      5.6.1. Support more than one NETCONF client ............ 51
      5.6.2. Support NETCONF clients having more than one endpoint 51
      5.6.3. Support a reconnection strategy .................... 51
      5.6.4. Support both persistent and periodic connections ... 52
      5.6.5. Reconnection strategy for periodic connections ... 52
      5.6.6. Keep-alives for persistent connections ............ 52
      5.6.7. Customizations for periodic connections .......... 52
6. Security Considerations ....................................... 52
7. IANA Considerations .......................................... 53
   7.1. The IETF XML Registry .................................. 53
   7.2. The YANG Module Names Registry ......................... 54
8. References .................................................... 54
   8.1. Normative References .................................... 54
   8.2. Informative References .................................. 55
Appendix A. Change Log ............................................ 57
   A.1. 00 to 01 .................................................. 57
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.

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.

3.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-client" module. Just the container is displayed below, but there is also a reusable grouping called "netconf-client-grouping" that the container is using.

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

module: ietf-netconf-client
  +--rw netconf-client
    +--rw initiate! {initiate}?
      |   +--rw netconf-server* [name]
      |     +--rw name                  string
      |     +--rw endpoints
      |        +--rw endpoint* [name]
      |        |   +--rw name         string
      |        |   +--rw (transport)
      |        |     +--:(ssh) {ssh-initiate}?
      |        |     +--rw ssh
      |        |     |   +--rw remote-address          inet:host
      |        |     |   +--rw remote-port?          inet:port-number
      |        |     |     |   +--rw local-address?          inet:ip-addr
      |        |     |     |     |   +--rw local-port?          inet:port-number
      |        |     |     |     |     |   +--rw tcp-keepalives {tcp-client-keepalive}
      |        |     |     |     |     |     |   +--rw idle-time?        uint16
      |        |     |     |     |     |     |   +--rw max-probes?       uint16
      |        |     |     |     |     |     |   +--rw probe-interval?   uint16
      |        |     |     |     |     |     |     |   +--rw ssh-client-identity
      |        |     |     |     |     |     |     |     |   +--rw username?            string
      |        |     |     |     |     |     |     |     |     |   +--rw (auth-type)
      |        |     |     |     |     |     |     |     |     |     |   +--:(password)
      |        |     |     |     |     |     |     |     |     |     |     |   +--rw password?      string

Watsen
Expires September 10, 2019

Internet-Draft
NETCONF Client and Server Models
March 2019
---rw keystore-reference?
  ks:asymmetric-key

---rw ssh-server-auth
  ---rw pinned-ssh-host-keys?
  ta:pinned-host-keys-ref
  {ta:ssh-host-keys}?
  ---rw pinned-ca-certs?
  ta:pinned-certificates-ref
  {sshcmn:ssh-x509-certs,ta:x509-}

---rw ssh-transport-params
  {ssh-client-transport-params-confi}

---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 ssh-keepalives {ssh-client-keepalive}?

---rw max-wait? uint16
  ---rw max-attempts? uint8

---:(tls) {tls-initiate}?
  ---rw tls
    ---rw remote-address inet:host
    ---rw remote-port? inet:port-num?

---rw local-address? inet:ip-address

---rw local-port? inet:port-num?

---rw tcp-keepalives {tcp-client-keepalive}?
\d}?
 | | | +--rw keystore-reference?
 | | | ks:asymmetric-key
\y-certificate-ref
 | | +--rw tls-server-auth
 | | | +--rw pinned-ca-certs?
 | | | ta:pinned-certificates-ref
 | | | {ta:x509-certificates}?
 | | | +--rw pinned-server-certs?
 | | | ta:pinned-certificates-ref
 | | | {ta:x509-certificates}?
 | | +--rw tls-hello-params
 | | | {tls-client-hello-params-config}?
 | | +--rw tls-versions
 | | | +--rw tls-version* identityref
 | | | +--rw cipher-suites
 | | | +--rw cipher-suite* identityref
 | | +--rw tls-keepalives {tls-client-keepalive}
\s}?
 | | +--rw max-wait? uint16
 | | +--rw max-attempts? uint8
 +--rw connection-type
 | +--rw (connection-type)
 | | +--:{(persistent-connection)
 | | | +--rw persistent!
 | | +--:(periodic-connection)
 | | | +--rw periodic!
 | | | | +--rw period? uint16
 | | | | +--rw anchor-time? yang:date-and-time
 | | | | +--rw idle-timeout? uint16
 | +--rw reconnect-strategy
 | | +--rw start-with? enumeration
 | | +--rw max-attempts? uint8
 +--rw listen! {listen}?
 | +--rw idle-timeout? uint16
 +--rw endpoint* [name]
 | +--rw name string
 +--rw (transport)
 | +--:{(ssh) {ssh-listen}?
 | | +--rw ssh
 | | | +--rw local-address inet:ip-address
 | | | +--rw local-port? inet:port-number
 | | | +--rw tcp-keepalives {tcp-server-keepalives}?
 | | | | +--rw idle-time? uint16
 | | | | +--rw max-probes? uint16
 | | | | +--rw probe-interval? uint16
++-rw ssh-client-identity
  |  +-rw username?          string

Watsen                 Expires September 10, 2019               [Page 9]
Internet-Draft      NETCONF Client and Server Models          March 2019

++-rw (auth-type)
  |  +--:(password)
  |    |  +--rw password?    string
  |  +--:(public-key)
  |    |  +--rw public-key
  |    |      +--rw (local-or-keystore)
  |    |      |  +--:(local) {local-keys-supported\}
  |    \}?
  |        |  +++-rw local-definition
  |        |      |  +++-rw algorithm?
  |        |      |      |  asymmetric-key-algo\}

\(y\)-algorithm-ref
  |  |  |  |  |  +++-x generate-hidden-key
  |  |  |  |  |      +--w input
  |  |  |  |  |      |  +++-w algorithm
  |  |  |  |  |      |      asymmetric-key-ke\

\(y\)-algorithm-ref
  |  |  |  |  |  +++-x install-hidden-key
  |  |  |  |  |      +--w input
  |  |  |  |  |      |  +++-w algorithm
  |  |  |  |  |      |      asymmetric-key-ke\

\(d\)?
  |  |  |  |  |  +++-rw keystore-reference?
  |  |  |  |  |      ks:asymmetric-key-ref
  |  |  |  |  |  +++-:(certificate)
  |  |  |  |  |      +++-rw certificate {sshcmn:ssh-x509-cert\}

\(s\)?
  |  |  |  |  |  +++-rw (local-or-keystore)
  |  |  |  |  |      +++-:(local) {local-keys-supported\}


\icates
|-- +--rw pinned-server-certs?
   |   --ta:pinned-certificates-ref
   |   --{sshcmn:ssh-x509-certs,ta:x509-certif\icates
|-- +--rw ssh-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 ssh-keepalives {ssh-client-keepalives}?

---(tls) {tls-listen}?
  +--rw tls
     +--rw local-address inet:ip-address
     +--rw local-port? inet:port-number
     +--rw tcp-keepalives {tcp-server-keepalives}?
     |   +--rw idle-time? uint16
     |   +--rw max-probes? uint16
     |   +--rw probe-interval? uint16
     +--rw tls-client-identity
     |   +--rw (auth-type)
     |     +--:(certificate)
     |     |   +--rw certificate
     |     |   |   +--rw (local-or-keystore)
     |     |   |     +--:(local) {local-keys-supported\rithm-ref
     |     |     +--rw local-definition
     |     |     |   +--rw algorithm?
     |     |     |     +--asymmetric-key-algo\rithm-ref
     |     |     |   +--rw public-key?
     |     |     |     |   binary
     |     |     |   +--rw private-key?
     |     |     |     |   union

Watsen                 Expires September 10, 2019              [Page 11]

Internet-Draft      NETCONF Client and Server Models          March 2019
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 3.2 of [I-D.ietf-netconf-keystore].

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

```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>
             <remote-address>corp-fw1.example.com</remote-address>
             <tcp-keepalives>
               <idle-time>15</idle-time>
               <max-probes>3</max-probes>
               <probe-interval>30</probe-interval>
             </tcp-keepalives>
           </ssh>
         </endpoint>
       </endpoints>
     </netconf-server>
   </initiate>

   <!-- SSH client identity -->
   <ssh-client-identity>
     <username>foobar</username>
     <public-key>
       <local-definition>
       </local-definition>
       <private-key>base64encodedvalue==</private-key>
       <public-key>base64encodedvalue==</public-key>
     </public-key>
   </ssh-client-identity>
</netconf-client>
```
<ssh-server-auth>
  <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinned-ca-certs>
  <pinned-server-certs>explicitly-trusted-server-certs</pinned-server-certs>
</ssh-server-auth>

<ssh-keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</ssh-keepalives>
</ssh>
</endpoint>

<endpoint>
  <name>corp-fw2.example.com</name>
  <ssh>
    <remote-address>corp-fw2.example.com</remote-address>
    <tcp-keepalives>
      <idle-time>15</idle-time>
      <max-probes>3</max-probes>
      <probe-interval>30</probe-interval>
    </tcp-keepalives>
    <ssh-client-identity>
      <username>foobar</username>
      <public-key>
        <local-definition>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
        </local-definition>
      </public-key>
    </ssh-client-identity>
    <ssh-server-auth>
      <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinned-ca-certs>
      <pinned-server-certs>explicitly-trusted-server-certs</pinned-server-certs>
    </ssh-server-auth>
  </ssh>
</endpoint>
<!-- endpoints to listen for NETCONF Call Home connections on -->
<listen>
  <endpoint>
    <name>Intranet-facing listener</name>
    <ssh>
      <local-address>192.0.2.7</local-address>
      <ssh-client-identity>
        <username>foobar</username>
        <public-key>
          <local-definition>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
          </local-definition>
        </public-key>
      </ssh-client-identity>
      <ssh-server-auth>
        <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinned-ca-certs>
        <pinned-server-certs>explicitly-trusted-server-certs</pinned-server-certs>
        <pinned-ssh-host-keys>explicitly-trusted-ssh-host-keys</pinned-ssh-host-keys>
      </ssh-server-auth>
    </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@2019-03-09.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;
    reference
      "RFC AAAA: YANG Groupings for TCP Clients and TCP Servers";
  }

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

  import ietf-ssh-client {
    prefix sshc;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC YYYY: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
</CODE>
This module contains a collection of YANG definitions for configuring NETCONF clients.

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

revision "2019-03-09" {
  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

}

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";
}

// Groupings

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

  container initiate {
    if-feature 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 initiate connections to in parallel.";
      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."
}

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."
      uses tcpc:tcp-client-grouping {
        refine "remote-port" {
          default 830;
          description
            "The NETCONF client will attempt to connect
            to the IANA-assigned well-known port value
            for 'netconf-ssh' (443) if no value is
            specified."
        }
      }
      uses sshc:ssh-client-grouping;
    } // container ssh
  } // case ssh
  case tls {
    if-feature tls-initiate;
    container tls {
      description
        "Specifies IP and TLS specific configuration
         for the connection."
      uses tcpc:tcp-client-grouping {
        refine "remote-port" {
          default 6513;
          description
        }
      }
      uses sshc:ssh-client-grouping;
    } // container ssh
  } // case tls
"The NETCONF client will attempt to connect to the IANA-assigned well-known port value for 'netconf-tls' (6513) if no value is specified."

uses tlsc:tls-client-grouping {
refine "tls-client-identity/auth-type" {
mandatory true;
description
"NETCONF/TLS clients MUST pass some authentication credentials.";
}
}

} // container tls
} // case tls

} // choice transport
} // 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 it, using the reconnection strategy.";

} // container persistent
} // case persistent-connection
} // container connection-type

This connection type minimizes any NETCONF server to NETCONF client data-transfer delay, albeit at the expense of holding resources longer.

```yml
}]
}
case periodic-connection {
  container periodic {
    presence
    "Indicates that a periodic connection is to be maintained.";
    description
    "Periodically connect to the NETCONF server. The NETCONF server should close the connection upon completing planned activities.

    This connection type increases resource utilization, albeit with increased delay in NETCONF server to NETCONF client interactions."
    leaf period {
      type uint16;
      units "minutes";
      default 60;
      description
      "Duration of time between periodic connections.";
    }
    leaf anchor-time {
      type yang:date-and-time {
        // constrained to minute-level granularity
        pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:'
        + '(Z|[-+\-]\d{2}:\d{2})';
      }
      description
      "Designates a timestamp before or after which a series of periodic connections are determined. The periodic connections occur at a whole multiple interval from the anchor time. For example, for an anchor time is 15 minutes past midnight and a period interval of 24 hours, then a periodic connection will occur 15 minutes past midnight everyday.";
    }
  }
```
leaf idle-timeout {
    type uint16;
    units "seconds";
    default 120; // two minutes
    description
        "Specifies the maximum number of seconds that
         a NETCONF session may remain idle. A NETCONF
         session will be dropped if it is idle for an
         interval longer than this number of seconds.
         If set to zero, then the NETCONF client will
         never drop a session because it is idle.";
}

container reconnect-strategy {
    description
        "The reconnection strategy directs how a NETCONF client
        reconnects to a NETCONF server, after discovering its
        connection to the server has dropped, even if due to a
        reboot. The NETCONF client starts with the specified
        endpoint and tries to connect to it max-attempts times
        before trying the next endpoint in the list (round
        robin)."
    leaf start-with {
        type enumeration {
            enum first-listed {
                description
                    "Indicates that reconnections should start with
                    the first endpoint listed.";
            }
            enum last-connected {
                description
                    "Indicates that reconnections should start with
                    the endpoint last connected to. If no previous
                    connection has ever been established, then the
                    first endpoint configured is used. NETCONF
                    clients SHOULD be able to remember the last
                    endpoint connected to across reboots.";
            }
            enum random-selection {

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

container listen {
  if-feature 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 {
  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.";
      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.";
        }
      }
      uses sshc:ssh-client-grouping;
    }
  }
  case tls {
    if-feature tls-listen;
    container tls {
      description
        "TLS-specific listening configuration for inbound
         connections.";
      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.";
        }
      }
    }
  }
}
4. 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.

4.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-server" module. Just the container is displayed below, but there is also a reusable grouping called "netconf-server-grouping" that the container is using.

========== NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========
Watsen
Expires September 10, 2019

Internet-Draft NETCONF Client and Server Models March 2019
---rw tcp-keepalives {tcp-client-keepalive\s}?---rw idle-time? uint16
---rw max-probes? uint16
---rw probe-interval? uint16
---rw ssh-server-identity
  ---rw host-key* [name]
    ---rw name string
    ---rw (host-key-type)
      ---:(public-key)
        ---rw public-key
        ---:(local-or-keystore)
          ---:(local)
            ---:{local-keys-supported}
        ---:(public-key)
        ---:(private-key)
          ---:{local-keys-supported}
    ---:{local-keys-supported}

---rw (host-key-type)
  ---:(public-key)
    ---rw public-key
    ---:(local-or-keystore)
      ---:(local)
        ---:{local-keys-supported}
       ---:{local-keys-supported}

---rw public-key?
  ---:binary
  ---:union

---rw private-key?
  ---:binary
  ---:union

----x generate-hidden-key

----x install-hidden-key

---w input

---w algorithm?
  ---:asymmetric

---w input

---w algorithm?
  ---:asymmetric

---w public-key
  ---:binary

---w private-key
  ---:binary

----x generate-hidden-key

----x install-hidden-key

---w input

---w algorithm?
  ---:asymmetric

---w public-key
  ---:binary

---w private-key
  ---:binary

----:(keystore)
  ---:{keystore-supported}
| \nce?          | \--rw keystore-reference |
| \-key-ref     | \ks:asymmetric |
| \}?           | \--rw (local-or-keystore) |
| \|?           | \--rw local-definition |
| \-key-algorithm-ref | \--rw public-key? |
| \den-key      | \--x generate-hidden |
| \thm          | \--x install-hidden |
| \metric-key-algorithm-ref | \--w input |
| \en-key       | \--w input |
| \thm          | \--w asymmetric |
| \metric-key-algorithm-ref | \--w public |
| \-key?        | \bina |
| \ry           | \--w private |
| \e-key?       | \bina |
| \ry           | \--rw certificate |
---(keystore)
{keystore-supported}
---rw keystore-refere

---rw ssh-transport-params
{ssh-server-transport-params-confi}

---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 ssh-keepalives {ssh-server-keepalive}

---rw max-wait? uint16
| | +--rw max-attempts? uint8

---(tls) {tls-call-home}?
| | +--rw tls
| | +--rw remote-address inet:host
| +--rw remote-port?         inet:port-num |
| +--rw local-address?       inet:ip-address |
| +--rw local-port?          inet:port-num |
| +--rw tcp-keepalives {tcp-client-keepalive}?
|    | +--rw idle-time?           uint16 |
|    | +--rw max-probes?          uint16 |
|    | +--rw probe-interval?      uint16 |
|    | +--rw tls-server-identity  |

Watsen Expires September 10, 2019 [Page 32]

Internet-Draft NETCONF Client and Server Models March 2019

| +--rw (local-or-keystore)
|    +--:(local) {local-keys-supported}?
|    | +--rw local-definition
|    |    +--rw algorithm?
|    |    | +-- asymmetric-key-algorithm-ref |
|    | +--rw public-key?
|    |    binary |
|    | +--rw private-key?
|    |    union |
|    |    +--x generate-hidden-key
|    |    | +--w input
|    |    |    +--w algorithm
|    |    |    | +-- asymmetric-key-algorithm-ref |
|    | +--x install-hidden-key
|    |    +--w input
|    |    | +--w algorithm
|    |    |    | +-- asymmetric-key-algorithm-ref |

algorithm-ref

| +--w public-key?  bin |
| +--w private-key? bin |

ary

| +--rw cert?
|    | +-- end-entity-cert-cms |
|    |    +--n certificate-expiration
|    |    | +-- expiration-date
|    |    |    | +-- yang:date-and-time
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 3.2 of [I-D.ietf-netconf-keystore].

========== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) ===========
<endpoint> <!-- listening for TLS sessions -->
  <name>netconf/tls</name>
  <tls>
    <local-address>192.0.2.7</local-address>
    <tls-server-identity>
      <local-definition>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </tls-server-identity>
    <tls-client-auth>
      <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-certs>
      <pinned-client-certs>explicitly-trusted-client-certs</pinned-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>
    </tls-client-auth>
  </tls>
</endpoint>

<!-- calling home to SSH and TLS based NETCONF clients -->
<call-home>
  <netconf-client> <!-- SSH-based client -->
  </netconf-client>
</call-home>
<name>config-mgr</name>
<endpoints>
  <endpoint>
    <name>east-data-center</name>
    <ssh>
      <remote-address>east.config-mgr.example.com</remote-address>
      <ssh-server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <local-definition>
              <private-key>base64encodedvalue==</private-key>
              <public-key>base64encodedvalue==</public-key>
            </local-definition>
          </public-key>
        </host-key>
      </ssh-server-identity>
      <ssh-client-cert-auth>
        <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-certs>
        <pinned-client-certs>explicitly-trusted-client-certs</pinned-client-certs>
      </ssh-client-cert-auth>
    </ssh>
  </endpoint>
  <endpoint>
    <name>west-data-center</name>
    <ssh>
      <remote-address>west.config-mgr.example.com</remote-address>
      <ssh-server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <local-definition>
              <private-key>base64encodedvalue==</private-key>
              <public-key>base64encodedvalue==</public-key>
            </local-definition>
          </public-key>
        </host-key>
      </ssh-server-identity>
    </ssh>
  </endpoint>
</endpoints>
<public-key>base64encodedvalue==</public-key>
</local-definition>
</host-key>
</ssh-server-identity>
<ssh-client-cert-auth>
<pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-certs>
<pinned-client-certs>explicitly-trusted-client-certs</pinned-client-certs>
</ssh-client-cert-auth>
</ssh>
</endpoint>
</endpoints>
<connection-type>
<periodic>
<idle-timeout>300</idle-timeout>
<period>60</period>
</periodic>
</connection-type>
<reconnect-strategy>
<start-with>last-connected</start-with>
<max-attempts>3</max-attempts>
</reconnect-strategy>
</netconf-client>
<netconf-client> <!-- TLS-based client -->
<name>data-collector</name>
<endpoints>
<endpoint>
<name>east-data-center</name>
<tls>
<remote-address>east.analytics.example.com</remote-address>
</tls>
</endpoint>
</endpoints>
<local-definition>
<private-key>base64encodedvalue==</private-key>
<public-key>base64encodedvalue==</public-key>
<cert>base64encodedvalue==</cert>
</local-definition>
</tls-server-identity>
</tls-client-auth>
<pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-certs>
<pinned-client-certs>explicitly-trusted-client-certs</pinned-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>
</tls-client-auth>
<tcp-keepalives>
  <idle-time>15</idle-time>
  <max-probes>3</max-probes>
  <probe-interval>30</probe-interval>
</tcp-keepalives>
<tls-keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</tls-keepalives>
</tls>
</endpoint>
<endpoint>
  <name>west-data-center</name>
  <tls>
    <remote-address>west.analytics.example.com</remote-address>
    <tls-server-identity>
      <local-definition>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </tls-server-identity>
  </tls-client-auth>
  <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-certs>
  <pinned-client-certs>explicitly-trusted-client-certs</pinned-client-certs>
  <cert-maps>
    <cert-to-name>
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
This YANG module imports YANG types from [RFC6991], and YANG groupings from [RFC7407], [I-D.ietf-netconf-ssh-client-server] and [I-D.ietf-netconf-ssh-client-server].

<CODE BEGINS> file "ietf-netconf-server@2019-03-09.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 AAAA: YANG Groupings for TCP Clients and TCP Servers";
  }

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

  import ietf-ssh-server {
    prefix sshs;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC YYYYY: YANG Groupings for SSH Clients and SSH Servers";

import ietf-tls-server {
    prefix tlss;
    revision-date 2019-03-09; // stable grouping definitions
    reference
        "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
}

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

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

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

    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.

    Copyright (c) 2019 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD
    License set forth in Section 4.c of the IETF Trust's
    Legal Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info)."
revision "2019-03-09" {
  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";
}

// Groupings

grouping netconf-server-grouping {

description
  "Top-level grouping for NETCONF server configuration."
container listen {
  if-feature listen;
  presence "Enables server to listen for TCP connections";
  description "Configures listen behavior";
  leaf idle-timeout {
    type uint16;
    units "seconds";
    default 3600; // one hour
  }
}
"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."
  }
  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."
        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 {
      uses sshs:ssh-server-grouping;
    }
  }
}
if-feature tls-listen;
container tls {
    description
        "TLS-specific listening configuration for inbound
        connections.";
    uses tcps:tcp-server-grouping {
        refine "local-port" {
            default 6513;
            description
                "The NETCONF server will listen on the IANA-
                assigned well-known port value for 'netconf-tls'
                (6513) if no value is specified.";
        }
    }
    uses tlss:tls-server-grouping {
        refine "tls-client-auth" {
            must 'pinned-ca-certs or pinned-client-certs';
            description
                "NETCONF/TLS servers MUST validate client
                certificiates.";
        }
    }
    augment "tls-client-auth" {
        description
            "Augments in the cert-to-name structure.";
        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 WWWW: NETCONF over TLS, Section 7";
        }
    }
}
container call-home {
    if-feature call-home;
    presence "Enables server to initiate TCP connections";
    description "Configures call-home behavior";
    list netconf-client {
        key name;
        min-elements 1;
        description
            "List of NETCONF clients the NETCONF server is to
             initiate call-home connections to in parallel.";
        leaf name {
            type string;
            description
                "An arbitrary name for the remote NETCONF client.";
        }
    }
    container endpoints {
        description
            "Container for the list of endpoints.";
        list endpoint {
            key name;
            min-elements 1;
            ordered-by user;
            description
                "A non-empty user-ordered list of endpoints for this
                 NETCONF server to try to connect to in sequence.
                 Defining more than one enables high-availability.";
            leaf name {
                type string;
                description
                    "An arbitrary name for this endpoint.";
            }
        }
        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 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."

uses sshs:ssh-server-grouping;
}
}

} uses tlss:tls-server-grouping {
refine "tls-client-auth" {
must 'pinned-ca-certs or pinned-client-certs';
description
"NETCONF/TLS servers MUST validate client certificates.";
}
augment "tls-client-auth" { 
description
"Augments in the cert-to-name structure.";
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 WWWW: NETCONF over TLS, Section 7";
case periodic-connection {
  container periodic {
    presence
      "Indicates that a periodic connection is to be maintained.";
    description
      "Periodically connect to the NETCONF client. The NETCONF client should close the underlying TLS connection upon completing planned activities.

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

} // container periodic
} // 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 {

desc
default first-listed;
description
"Specifies which of the NETCONF client's endpoints
the NETCONF server should start with when trying
to connect to the NETCONF client."
}

leaf max-attempts {
  type uint8 {
    range "1..max";
  }
  default 3;
description
"Specifies the number times the NETCONF server tries
to connect to a specific endpoint before moving on
to the next endpoint in the list (round robin)."
}

} // container reconnect-strategy
} // list netconf-client
} // container call-home
} // grouping netconf-server-grouping

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

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

5.1. 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.

5.2. 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.

5.3. 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.

5.4. 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
5.5. 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.

5.6. For Call Home connections

The following objectives only pertain to call home connections.

5.6.1. 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.

5.6.2. 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.

5.6.3. 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 of 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.
5.6.4. 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.

5.6.5. 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.

5.6.6. 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.

5.6.7. 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).

6. Security Considerations

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) [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 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:

/:

The entire data trees defined by the modules defined in this draft are sensitive to write operations. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., can dramatically alter the implemented security policy. However, no NACM annotations are applied as the data SHOULD be editable by users other than a designated 'recovery session'.

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

7. IANA Considerations
7.1. 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:

Internet-Draft NETCONF Client and Server Models March 2019

Watsen Expires September 10, 2019

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.

7.2. 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

8. References

8.1. Normative References

[I-D.ietf-netconf-keystore]

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., Wu, G., and L. Xia, "YANG Groupings for SSH

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

[I-D.kwatsen-netconf-tcp-client-server]
Watsen, K., "YANG Groupings for TCP Clients and TCP Servers", draft-kwatsen-netconf-tcp-client-server-00 (work in progress), March 2019.

Watsen                 Expires September 10, 2019              [Page 54]
Internet-Draft      NETCONF Client and Server Models          March 2019

[RFC2119]  Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,

DOI 10.17487/RFC6020, October 2010,

DOI 10.17487/RFC6241, June 2011,

[RFC6242]  Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242,
DOI 10.17487/RFC6242, June 2011,

DOI 10.17487/RFC6991, July 2013,

DOI 10.17487/RFC7407, December 2014,


8.2. Informative References


Appendix A. Change Log

A.1. 00 to 01

  o Renamed "keychain" to "keystore".

A.2. 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.
Updated both modules to accommodate new groupings in the ssh/tls drafts.

A.3. 02 to 03

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

A.4. 03 to 04

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

A.5. 04 to 05

- Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- Updated examples to inline key and certificates (no longer a leafref to keystore)

A.6. 05 to 06

- Fixed change log missing section issue.
- Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.
Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

- Removed "idle-timeout" from "persistent" connection config.
- Added "random-selection" for reconnection-strategy's "starts-with" enum.
- Replaced "connection-type" choice default (persistent) with "mandatory true".
- Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.
- Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

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

Appendix B. 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.

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