

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 6, 2018

K. Watsen
Juniper Networks
G. Wu
Cisco Networks
June 4, 2018

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-06

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 "2018-06-04" --> 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 <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 December 6, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	3
3. The NETCONF Client Model	4
3.1. Tree Diagram	4
3.2. Example Usage	9
3.3. YANG Module	11
4. The NETCONF Server Model	22
4.1. Tree Diagram	22

4.2.	Example Usage	28
4.3.	YANG Module	33
5.	Design Considerations	44
5.1.	Support all NETCONF transports	45
5.2.	Enable each transport to select which keys to use	45
5.3.	Support authenticating NETCONF clients certificates	45
5.4.	Support mapping authenticated NETCONF client certificates to usernames	45
5.5.	Support both listening for connections and call home	45
5.6.	For Call Home connections	46
5.6.1.	Support more than one NETCONF client	46
5.6.2.	Support NETCONF clients having more than one endpoint	46
5.6.3.	Support a reconnection strategy	46
5.6.4.	Support both persistent and periodic connections	46
5.6.5.	Reconnection strategy for periodic connections	46
5.6.6.	Keep-alives for persistent connections	47
5.6.7.	Customizations for periodic connections	47
6.	Security Considerations	47
7.	IANA Considerations	48
7.1.	The IETF XML Registry	48
7.2.	The YANG Module Names Registry	48
8.	References	49
8.1.	Normative References	49
8.2.	Informative References	50
Appendix A.	Change Log	51
A.1.	00 to 01	51
A.2.	01 to 02	51
A.3.	02 to 03	51
A.4.	03 to 04	51
A.5.	04 to 05	51
A.6.	05 to 06	52
Acknowledgements	52
Authors' Addresses	52

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 by the same name that the container is using.

[Note: '\' line wrapping for formatting only]

```

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 address? inet:host
                  +--rw port? inet:port-number\
                +--rw client-identity
                  +--rw username? string
                  +--rw (auth-type)
                    +--:(password)

```



```

|--rw server-auth
|   |--rw pinned-ca-certs?
|   |   ta:pinned-certificates-ref
|   |--rw pinned-server-certs?
|   |   ta:pinned-certificates-ref
|--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 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 client-identity
|   |   |   |--rw username?         string
|   |   |   |--rw (auth-type)
|   |   |   |   +--:(password)
|   |   |   |   |   |--rw password?    string
|   |   |   |   +--:(public-key)
|   |   |   |   |   |--rw public-key
|   |   |   |   |   |   |--rw (local-or-keystore)
|   |   |   |   |   |   |   +--:(local)
|   |   |   |   |   |   |   |   |--rw algorithm
|   |   |   |   |   |   |   |   |   ct:key-algorithm-ref
|   |   |   |   |   |   |   |--rw public-key    binary
|   |   |   |   |   |   |--rw private-key     union

```



```

    +--rw port?                inet:port-number
    +--rw client-identity
    |   +--rw (auth-type)
    |   |   +--:(certificate)
    |   |   |   +--rw certificate
    |   |   |   |   +--rw (local-or-keystore)
    |   |   |   |   |   +--:(local)
    |   |   |   |   |   |   +--rw algorithm
    |   |   |   |   |   |   |   ct:key-algorithm-ref
    |   |   |   |   |   |   +--rw public-key
    |   |   |   |   |   |   |   binary
    |   |   |   |   |   |   +--rw private-key
    |   |   |   |   |   |   |   union
    |   |   |   |   |   |   +--rw cert
    |   |   |   |   |   |   |   ct:end-entity-cert-cms\
    |   |   |   |   |   |   |
    |   |   |   |   |   |   +---n certificate-expiration
    |   |   |   |   |   |   |   +-- expiration-date?
    |   |   |   |   |   |   |   |   yang:date-and-time
    |   |   |   |   |   |   +--:(keystore)
    |   |   |   |   |   |   |   {keystore-implemented}?
    |   |   |   |   |   |   +--rw reference
    |   |   |   |   |   |   |   ks:asymmetric-key-cert\
    |   |   |   |   |   |   |
    |   |   |   |   |   |   certificate-ref
    |   |   |   |   |   |
    +--rw server-auth
    |   +--rw pinned-ca-certs?
    |   |   ta:pinned-certificates-ref
    |   +--rw pinned-server-certs?
    |   |   ta:pinned-certificates-ref
    +--rw hello-params
    |   {tls-client-hello-params-config}?
    |   +--rw tls-versions
    |   |   +--rw tls-version*  identityref
    |   +--rw cipher-suites
    |   |   +--rw cipher-suite*  identityref

```

3.2. Example Usage

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

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

[Note: '\ ' line wrapping for formatting only]

```

<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>
            <address>corp-fw1.example.com</address>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
f-crypto-types">ct:secp521r1</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
              </public-key>
            </client-identity>
            <server-auth>
              <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
inned-ca-certs>
              <pinned-server-certs>explicitly-trusted-server-certs</\
pinned-server-certs>
            </server-auth>
          </ssh>
        </endpoint>
        <endpoint>
          <name>corp-fw2.example.com</name>
          <ssh>
            <address>corp-fw2.example.com</address>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
f-crypto-types">ct:secp521r1</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
              </public-key>
            </client-identity>
            <server-auth>
              <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
inned-ca-certs>
              <pinned-server-certs>explicitly-trusted-server-certs</\
pinned-server-certs>
            </server-auth>
          </ssh>
        </endpoint>
      </endpoints>
    </netconf-server>
  </initiate>
</netconf-client>

```

```

        </ssh>
    </endpoint>
</endpoints>
<connection-type>
  <persistent/>
</connection-type>
<reconnect-strategy>
  <start-with>last-connected</start-with>
</reconnect-strategy>
</netconf-server>
</initiate>

<!-- endpoints to listen for NETCONF Call Home connections on -->
<listen>
  <endpoint>
    <name>Intranet-facing listener</name>
    <ssh>
      <address>192.0.2.7</address>
      <client-identity>
        <username>foobar</username>
        <public-key>
          <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cr\
ypto-types">ct:secp521r1</algorithm>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
        </public-key>
      </client-identity>
      <server-auth>
        <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinne\
d-ca-certs>
        <pinned-server-certs>explicitly-trusted-server-certs</pinn\
ed-server-certs>
        <pinned-ssh-host-keys>explicitly-trusted-ssh-host-keys</pi\
nned-ssh-host-keys>
      </server-auth>
    </ssh>
  </endpoint>
</listen>
</netconf-client>

```

3.3. YANG Module

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

```

<CODE BEGINS> file "ietf-netconf-client@2018-06-04.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 2018-06-04; // stable grouping definitions
  reference
    "RFC YYYY: YANG Groupings for SSH Clients and SSH Servers";
}

import ietf-tls-client {
  prefix ts;
  revision-date 2018-06-04; // 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: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) 2017 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 "2018-06-04" {
  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;
            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.";
        leaf address {
          type inet:host;
          description
            "The IP address or hostname of the endpoint.
            If a domain name is configured, then the
            DNS resolution should happen on each usage
            attempt. If the DNS resolution results in
            multiple IP addresses, the IP addresses will
            be tried according to local preference order
            until a connection has been established or
            until all IP addresses have failed.";
        }
        leaf port {
          type inet:port-number;
          default 830;
          description
            "The IP port for this endpoint. The NETCONF
            client will use the IANA-assigned well-known
            port for 'netconf-ssh' (830) if no value is
            specified.";
        }
      }
    }
  }
  uses ss:ssh-client-grouping;
```

```

    }
  } // end ssh
case tls {
  if-feature tls-initiate;
  container tls {
    description
      "Specifies IP and TLS specific configuration
      for the connection.";
    leaf address {
      type inet:host;
      description
        "The IP address or hostname of the endpoint.
        If a domain name is configured, then the
        DNS resolution should happen on each usage
        attempt. If the DNS resolution results in
        multiple IP addresses, the IP addresses will
        be tried according to local preference order
        until a connection has been established or
        until all IP addresses have failed.";
    }
    leaf port {
      type inet:port-number;
      default 6513;
      description
        "The IP port for this endpoint. The NETCONF
        client will use the IANA-assigned well-
        known port for 'netconf-tls' (6513) if no
        value is specified.";
    }
    uses ts:tls-client-grouping {
      refine "client-identity/auth-type" {
        mandatory true;
        description
          "NETCONF/TLS clients MUST pass some
          authentication credentials.";
      }
    }
  } // end tls
}
}
}

container connection-type {
  description
    "Indicates the kind of connection to use.";
  choice connection-type {
    default persistent-connection;
  }
}

```

```
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.

      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 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
```



```
        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;
    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.";
  leaf address {
    type inet:ip-address;
    description
      "The IP address to listen on for incoming call-
      home connections. The NETCONF client will listen
      on all configured interfaces if no value is
      specified. INADDR_ANY (0.0.0.0) or INADDR6_ANY
      (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be used when
      the server is to listen on all IPv4 or IPv6
      addresses, respectively.";
  }
  leaf port {
    type inet:port-number;
    default 4334;
    description
      "The port number to listen on for call-home
      connections. The NETCONF client will listen
      on the IANA-assigned well-known port for
      'netconf-ch-ssh' (4334) if no value is
      specified.";
  }
  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 on for incoming call-
        home connections. The NETCONF client will listen
        on all configured interfaces if no value is
        specified. INADDR_ANY (0.0.0.0) or INADDR6_ANY
        (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be used when
        the server is to listen on all IPv4 or IPv6
        addresses, respectively.";
    }
    leaf port {
      type inet:port-number;
    }
  }
}
```

```

    default 4335;
    description
      "The port number to listen on for call-home
      connections.  The NETCONF client will listen
      on the IANA-assigned well-known port for
      'netconf-ch-tls' (4335) if no value is
      specified.";
  }
  uses ts:tls-client-grouping {
    refine "client-identity/auth-type" {
      mandatory true;
      description
        "NETCONF/TLS clients MUST pass some
        authentication credentials.";
    }
  }
} // end transport
} // end endpoint
} // end listen

} // end netconf-client
}
<CODE ENDS>

```

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 by the same name that the container is using.

[Note: '\' line wrapping for formatting only]

```

module: ietf-netconf-server
  +--rw netconf-server
    +--rw listen! {listen}?
      +--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-identity
                +--rw host-key* [name]
                  +--rw name          string
                  +--rw (host-key-type)
                    +---:(public-key)
                      +--rw public-key
                        +--rw (local-or-keystore)
                          +---:(local)
                            +--rw algorithm
                              |
                              | ct:key-algorithm-re\
f          |
          | | +--rw public-key      binary\
          | | +--rw private-key    union
          | | +---:(keystore)
          | |   {keystore-implemented\
}?)
          | |   +--rw reference
          | |     ks:asymmetric-key-r\
ef
          +---:(certificate)
            +--rw certificate
              {sshcmn:ssh-x509-certs}?
            +--rw (local-or-keystore)
              +---:(local)
                +--rw algorithm
                  |
                  | ct:key-algorithm-re\
f          |
          | | +--rw public-key
          | | | binary
          | | +--rw private-key

```



```

ef | | | | ks:asymmetric-key-certificate-r\
| | | |
| | | | +--rw client-auth
| | | | | +--rw pinned-ca-certs?
| | | | | | ta:pinned-certificates-ref
| | | | | +--rw pinned-client-certs?
| | | | | | ta:pinned-certificates-ref
| | | | | +--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 endpoints
| +--rw endpoint* [name]
| | +--rw name string
| | +--rw (transport)
| | | +--:(ssh) {ssh-call-home}?
| | | | +--rw ssh
| | | | | +--rw address inet:host
| | | | | +--rw port? inet:port-number\
| | | |
| | | | +--rw server-identity
| | | | | +--rw host-key* [name]
| | | | | | +--rw name string
| | | | | | +--rw (host-key-type)
| | | | | | | +--:(public-key)
| | | | | | | | +--rw public-key
| | | | | | | | | +--rw (local-or-keystore)
| | | | | | | | | | +--:(local)
| | | | | | | | | | | +--rw algorithm
| | | | | | | | | | | | ct:key-algori\
| | | | | | | | | |
| | | | | | | | | | +--rw public-key
| | | | | | | | | | | binary
| | | | | | | | | | +--rw private-key
| | | | | | | | | | | union
| | | | | | | | | | | +--:(keystore)
thm-ref | | | |

```



```

|         | +--rw encryption-alg*  identityref
|         | +--rw mac
|         | +--rw mac-alg*  identityref
+---:(tls) {tls-call-home}?
+--rw tls
+--rw address          inet:host
+--rw port?           inet:port-number
+--rw server-identity
|   +--rw (local-or-keystore)
|   +---:(local)
|   |   +--rw algorithm
|   |   |   ct:key-algorithm-ref
|   |   +--rw public-key
|   |   |   binary
|   |   +--rw private-key
|   |   |   union
|   |   +--rw cert
|   |   |   ct:end-entity-cert-cms
|   |   +---n certificate-expiration
|   |   |   +-- expiration-date?
|   |   |   |   yang:date-and-time
|   |   +---:(keystore) {keystore-implemented\
|   |
|   |   +--rw reference
|   |   |   ks:asymmetric-key-certifi\
|   |
|   +---rw client-auth
|   |   +--rw pinned-ca-certs?
|   |   |   ta:pinned-certificates-ref
|   |   +--rw pinned-client-certs?
|   |   |   ta:pinned-certificates-ref
|   |   +--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!

```



```

        <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinne\
d-ca-certs>
        <pinned-client-certs>explicitly-trusted-client-certs</pinn\
ed-client-certs>
        </client-cert-auth>
    </ssh>
</endpoint>
<endpoint> <!-- listening for TLS sessions -->
    <name>netconf/tls</name>
    <tls>
        <address>192.0.2.7</address>
        <server-identity>
            <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cryp\
to-types">ct:secp521r1</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
            <cert>base64encodedvalue==</cert>
        </server-identity>
        <client-auth>
            <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinne\
d-ca-certs>
            <pinned-client-certs>explicitly-trusted-client-certs</pinn\
ed-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 SSH and TLS based NETCONF clients -->
<call-home>
    <netconf-client> <!-- SSH-based client -->
        <name>config-mgr</name>
        <endpoints>
            <endpoint>
                <name>east-data-center</name>

```

```

    <ssh>
      <address>east.config-mgr.example.com</address>
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\
ietf-crypto-types">ct:secp521r1</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
        </host-key>
      </server-identity>
      <client-cert-auth>
        <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
inned-ca-certs>
        <pinned-client-certs>explicitly-trusted-client-certs</\
pinned-client-certs>
      </client-cert-auth>
    </ssh>
  </endpoint>
  <endpoint>
    <name>west-data-center</name>
    <ssh>
      <address>west.config-mgr.example.com</address>
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\
ietf-crypto-types">ct:secp521r1</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
          </public-key>
        </host-key>
      </server-identity>
      <client-cert-auth>
        <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
inned-ca-certs>
        <pinned-client-certs>explicitly-trusted-client-certs</\
pinned-client-certs>
      </client-cert-auth>
    </ssh>
  </endpoint>
</endpoints>
<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>data-collector</name>
    <endpoints>
        <endpoint>
            <name>east-data-center</name>
            <tls>
                <address>east.analytics.example.com</address>
                <server-identity>
                    <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
crypto-types">ct:secp521r1</algorithm>
                    <private-key>base64encodedvalue==</private-key>
                    <public-key>base64encodedvalue==</public-key>
                    <cert>base64encodedvalue==</cert>
                </server-identity>
                <client-auth>
                    <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
inned-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>
                </client-auth>
            </tls>
        </endpoint>
        <endpoint>
            <name>west-data-center</name>
            <tls>
                <address>west.analytics.example.com</address>
                <server-identity>

```

```

        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
crypto-types">ct:secp521r1</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
    </server-identity>
    <client-auth>
        <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
inned-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>
    </client-auth>
</tls>
</endpoint>
</endpoints>
<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>

```

4.3. YANG Module

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

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@2018-06-04.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 2018-06-04; // stable grouping definitions
    reference
      "RFC YYYY: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-server {
    prefix ts;
    revision-date 2018-06-04; // 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:kwatsen@juniper.net>

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

Author: Juergen Schoenwaelder
        <mailto:j.schoenwaelder@jacobs-university.de>;
```

description

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

```
Copyright (c) 2017 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 "2018-06-04" {
  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 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
      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.";
        leaf address {
          type inet:ip-address;
        }
      }
    }
  }
}
```

```
mandatory true;
description
  "The IP address to listen on for incoming
  connections. The NETCONF server will listen
  on all configured interfaces if no value is
  specified. INADDR_ANY (0.0.0.0) or INADDR6_ANY
  (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be used when
  the server is to listen on all IPv4 or IPv6
  addresses, respectively.";
}
leaf port {
  type inet:port-number;
  default 830;
  description
    "The local port number to listen on. If no value
    is specified, the IANA-assigned port value for
    'netconf-ssh' (830) is used.";
}
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;
      mandatory true;
      description
        "The IP address to listen on for incoming
        connections. The NETCONF server will listen
        on all configured interfaces if no value is
        specified. INADDR_ANY (0.0.0.0) or INADDR6_ANY
        (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be used when
        the server is to listen on all IPv4 or IPv6
        addresses, respectively.";
    }
    leaf port {
      type inet:port-number;
      default 6513;
      description
        "The local port number to listen on. If no value
        is specified, the IANA-assigned port value for
        'netconf-tls' (6513) is used.";
    }
  }
  uses ts:tls-server-grouping {
```

```

    refine "client-auth" {
      must 'pinned-ca-certs or pinned-client-certs';
      description
        "NETCONF/TLS servers MUST validate client
        certificates.";
    }
    augment "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.";
        leaf address {
          type inet:host;
          mandatory true;
          description
            "The IP address or hostname of the endpoint.
            If a domain name is configured, then the
            DNS resolution should happen on each usage
            attempt. If the the DNS resolution results
            in multiple IP addresses, the IP addresses
            will be tried according to local preference
            order until a connection has been established
            or until all IP addresses have failed.";
        }
        leaf port {
          type inet:port-number;
          default 4334;
          description
            "The IP port for this endpoint. The NETCONF
            server will use the IANA-assigned well-known
            port for 'netconf-ch-ssh' (4334) if no value
            is specified.";
        }
      }
    }
  }
  uses ss:ssh-server-grouping;
}
```

```
}
case tls {
  if-feature tls-call-home;
  container tls {
    description
      "Specifies TLS-specific call-home transport
      configuration.";
    leaf address {
      type inet:host;
      mandatory true;
      description
        "The IP address or hostname of the endpoint.
        If a domain name is configured, then the
        DNS resolution should happen on each usage
        attempt. If the the DNS resolution results
        in multiple IP addresses, the IP addresses
        will be tried according to local preference
        order until a connection has been established
        or until all IP addresses have failed.";
    }
    leaf port {
      type inet:port-number;
      default 4335;
      description
        "The IP port for this endpoint. The NETCONF
        server will use the IANA-assigned well-known
        port for 'netconf-ch-tls' (4335) if no value
        is specified.";
    }
  }
  uses ts:tls-server-grouping {
    refine "client-auth" {
      must 'pinned-ca-certs or pinned-client-certs';
      description
        "NETCONF/TLS servers MUST validate client
        certificates.";
    }
    augment "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";
    }
}
}
} // end tls
} // end choice
} // end endpoint
}
container connection-type {
    description
        "Indicates the kind of connection to use.";
    choice connection-type {
        default persistent-connection;
        description
            "Selects between available connection types.";
        case persistent-connection {
            container persistent {
                presence
                    "Indicates that a persistent connection is to be
                    maintained.";
                description
                    "Maintain a persistent connection to the NETCONF
                    client. If the connection goes down, immediately
                    start trying to reconnect to 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 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.";
                }
            }
        }
    }
}

```


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 authenticated certificate. Thus the modules defined herein should enable this mapping to be configured.

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 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) [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:

/: 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 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.

7.2. 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
```

8. References

8.1. Normative References

- [I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a "Keystore" Mechanism",
draft-ietf-netconf-keystore-04 (work in progress), October
2017.
- [I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and
SSH Servers", draft-ietf-netconf-ssh-client-server-05
(work in progress), October 2017.
- [I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and
TLS Servers", draft-ietf-netconf-tls-client-server-05
(work in progress), October 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
and A. Bierman, Ed., "Network Configuration Protocol
(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
<<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure
Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,
<<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",
RFC 6991, DOI 10.17487/RFC6991, July 2013,
<<https://www.rfc-editor.org/info/rfc6991>>.

- [RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <<https://www.rfc-editor.org/info/rfc7407>>.
- [RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <<https://www.rfc-editor.org/info/rfc7589>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<https://www.rfc-editor.org/info/rfc8071>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

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.
- o Updated both modules to accomodate new groupings in the ssh/tls drafts.

A.3. 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.4. 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.

A.5. 04 to 05

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

A.6. 05 to 06

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

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, Juergen Schoenwaelder, Phil Shafer, Sean Turner, and Bert Wijnen.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Gary Wu
Cisco Networks

EMail: garywu@cisco.com