NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2015

K. Watsen
Juniper Networks
J. Schoenwaelder
Jacobs University Bremen

October 26, 2014

# NETCONF Server Configuration Model
# draft-ietf-netconf-server-model-04

## Abstract

This draft defines a NETCONF server configuration data model. This data model enables configuration of the NETCONF service itself, including which transports it supports, what ports they listen on, whether call-home is supported, and associated parameters.

## Status of This Memo

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

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

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

This Internet-Draft will expire on April 29, 2015.

Table of Contents

## 1. Introduction

This draft defines a NETCONF [RFC6241] server configuration data model. This data model enables configuration of the NETCONF service itself, including which transports are supported, what ports the server listens on, whether call-home is supported, and associated parameters.

### 1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

### 1.2. Tree Diagrams

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

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

## 2. Objectives

The primary purpose of the YANG module defined herein is to enable the configuration of the NETCONF server service on the device. This scope includes the following objectives:

### 2.1. Support all NETCONF transports

The YANG module should support all current NETCONF transports, namely NETCONF over SSH [RFC6242] and NETCONF over TLS [rfc5539bis], and be extensible to support future transports as necessary.

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

### 2.2. Enable each transport to select which keys to use

Systems may have a multiplicity of host-keys or server-certificates from which subsets are configured for specific uses. For instance, a system 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.

### 2.3. Support authenticating client-certificates

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

### 2.4. Support mapping authenticated client-certificates to usernames

Some transports (e.g., TLS) need additional support to map authenticated transport-level sessions to a NETCONF username. The NETCONF server model defined herein should define an ability for this mapping to be configured."

### 2.5. Support both Listening for connections and Call Home

NETCONF has always supported the server opening a port to listen for client connections. More recently the NETCONF working group defined support for call-home ([draft-ietf-netconf-call-home]). The module should configure both listening for connections and call-home.

Because implementations may not support both listening for connections and call home, YANG "feature" statements should be used so that implementation can accurately advertise the connection types it supports.

### 2.6. For Call Home connections

The following objectives only pertain to call home connections.

#### 2.6.1. Support more than one application

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

#### 2.6.2. Support applications having more than one server

An application managing a device may implement a high-availability strategy employing a multiplicity of active and/or passive servers. Therefore, when it is desired for a device to initiate call home connections, it should be able to connect to any of the application's servers.

#### 2.6.3. Support a reconnection strategy

Assuming an application has more than one server, then it becomes necessary to configure how a device should reconnect to the application should it lose its connection to the application's servers. Of primary interest is if the device should start with first server defined in a user-ordered list of servers or with the last server it was connected to. Secondary settings might specify the frequency of attempts and number of attempts per server. Therefore, a reconnection strategy should be configurable.

#### 2.6.4. Support both persistent and periodic connections

Applications may vary greatly on how frequently they need to interact with a device, how responsive interactions with devices need to be, and how many simultaneous connections they can support. Some applications may need a persistent connection to devices to optimize real-time interactions, while others are satisfied with periodic interactions and reduced resources required. Therefore, when it is necessary for devices to initiate connections, the type of connection desired should be configured.

#### 2.6.5. 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.

#### 2.6.6. 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

application requirements, and therefore keep-alive settings should be configurable on a per-application basis.

#### 2.6.7. Customizations for periodic connections

If a periodic connection is desired, it is necessary for the device to know how often it should connect. This delay essentially determines how long the application might have to wait to send data to the device. This setting does not constrain how often the device must wait to send data to the application, as the device should immediately connect to the application whenever it has data to send to it.

A common communication pattern is that one data transmission is many times closely followed by another. For instance, if the device needs to send a notification message, there's a high probability that it will send another shortly thereafter. Likewise, the application may have a sequence of pending messages to send. Thus, it should be possible for a device to hold a connection open until some amount of time of no data being transmitted as transpired.

## 3. Data Model

### 3.1. Overview

#### 3.1.1. The "session-options" subtree

```
module: ietf-netconf-server
   +--rw netconf-server
      +--rw session-options
         +--rw hello-timeout?   uint32
         +--rw idle-timeout?    uint32
```

The above subtree illustrates how this YANG module enables configuration of NETCONF session options, independent of any transport or connection strategy. Please see the YANG module (Section 3.2) for a complete description of these configuration knobs.

#### 3.1.2. The "listen" subtree

```
module: ietf-netconf-server
   +--rw netconf-server
      +--rw listen {"(ssh-listen or tls-listen)"}?  // YANG 1.1 syntax
         +--rw max-sessions?   uint16
         +--rw endpoint* [name]
            +--rw name           string
            +--rw (transport)
            |  +--:(ssh) {ssh-listen}?
            |  |  +--rw ssh
            |  |     +--rw address?     inet:ip-address
            |  |     +--rw port?        inet:port-number
            |  |     +--rw host-keys
            |  |        +--rw host-key*   string
            |  +--:(tls) {tls-listen}?
            |     +--rw tls
            |        +--rw address?        inet:ip-address
            |        +--rw port?           inet:port-number
            |        +--rw certificates
            |           +--rw certificate*   string
            +--rw keep-alives
               +--rw interval-secs?   uint8
               +--rw count-max?       uint8
```

The above subtree illustrates how this YANG module enables configuration for listening for remote connections, as described in [RFC6242] and [rfc5539bis]. Feature statements are used to limit both if listening is supported at all as well as for which transports. If listening for connections is supported, then the model enables configuring a list of listening endpoints, each configured with a user-specified name (the key field), the transport to use (i.e. SSH, TLS), and the IP address and port to listen on. The port field is optional, defaulting to the transport-specific port when not configured.

### 3.1.3. The "call-home" subtree

```
module: ietf-netconf-server
   +--rw netconf-server
      +--rw call-home {"(ssh-call-home or tls-call-home)"}? // YANG 1.1 syntax
         +--rw application* [name]
            +--rw name                  string
            +--rw (transport)
            |  +--:(ssh) {ssh-call-home}?
            |  |  +--rw ssh
            |  |     +--rw endpoints
            |  |     |  +--rw endpoint* [name]
            |  |     |     +--rw name       string
            |  |     |     +--rw address    inet:host
            |  |     |     +--rw port?      inet:port-number
            |  |     +--rw host-keys
            |  |        +--rw host-key*   string
            |  +--:(tls) {tls-call-home}?
            |     +--rw tls
            |        +--rw endpoints
            |        |  +--rw endpoint* [name]
            |        |     +--rw name       string
            |        |     +--rw address    inet:host
            |        |     +--rw port?      inet:port-number
            |        +--rw certificates
            |           +--rw certificate*   string
            +--rw connection-type
            |  +--rw (connection-type)?
            |     +--:(persistent-connection)
            |     |  +--rw persistent
            |     |     +--rw keep-alives
            |     |        +--rw interval-secs?   uint8
            |     |        +--rw count-max?       uint8
            |     +--:(periodic-connection)
            |        +--rw periodic
            |           +--rw timeout-mins?   uint8
            |           +--rw linger-secs?    uint8
            +--rw reconnect-strategy
               +--rw start-with?      enumeration
               +--rw interval-secs?   uint8
               +--rw count-max?       uint8
```

The above subtree illustrates how this YANG module enables configuration for call home, as described in [draft-ietf-netconf-call-home]. Feature statements are used to limit both if call-home is supported at all as well as for which transports, if it is. If call-home is supported, then the model supports configuring a list of applications to connect to. Each application is configured with a user-specified name (the key field), the transport to be used (i.e. SSH, TLS), and a list of remote

endpoints, each having a name, an IP address, and an optional port. Additionally, the configuration for each remote application indicates the connection-type (persistent vs. periodic) and associated parameters, as well as the reconnection strategy to use.

### 3.1.4. The "ssh" subtree

```
module: ietf-netconf-server
   +--rw netconf-server
      +--rw ssh
         +--ro host-keys
            +--ro host-key* [name]
               +--ro name                 string
               +--ro format-identifier    string
               +--ro data                 binary
               +--ro fingerprint          string
```

The above subtree illustrates how this YANG module provides SSH state independent of if the NETCONF server if listening or calling home. This data-model provides a read-only listing of currently configured TLC certificates.

### 3.1.5. The "tls" subtree

```
module: ietf-netconf-server
   +--rw netconf-server
      +--rw tls
         +--ro certificates
         |  +--ro certificate* [name]
         |     +--ro name    string
         |     +--ro data    binary
         +--rw client-auth
            +--rw trusted-ca-certs
            |  +--rw trusted-ca-cert*   binary
            +--rw trusted-client-certs
            |  +--rw trusted-client-cert*   binary
            +--rw cert-maps
               +--rw cert-to-name* [id]
                  +--rw id             uint32
                  +--rw fingerprint    x509c2n:tls-fingerprint
                  +--rw map-type       identityref
                  +--rw name           string
```

The above subtree illustrates how this YANG module provides TLS state and enables TLS configuration independent of if the NETCONF server if listening or calling home.  This data-model provides 1) a read-only listing of currently configured TLC certificates and 2) an ability to

configure how client-certificates are authenicated and how
authenticated client-certificates are mapped to NETCONF user names.

## 3.2. YANG Module

This YANG module imports YANG types from [RFC6991], and
[draft-ietf-netmod-snmp-cfg].

RFC Ed.: update the date below with the date of RFC publication
and remove this note.

<CODE BEGINS> file "ietf-netconf-server@2014-10-26.yang"

```
module ietf-netconf-server {

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

  import ietf-inet-types {
    prefix inet;                // RFC 6991
  }
  import ietf-x509-cert-to-name {
    prefix x509c2n;             // draft-ietf-netmod-snmp-cfg
  }

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

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

    WG Chair: Mehmet Ersue
              <mailto:mehmet.ersue@nsn.com>

    WG Chair: Bert Wijnen
              <mailto:bertietf@bwijnen.net>

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


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

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

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

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";
      // RFC Ed.: replace XXXX with actual RFC number and
      // remove this note

      // RFC Ed.: please update the date to the date of publication

      revision "2014-10-26" {  // YYYY-MM-DD
        description
         "Initial version";
        reference
         "RFC XXXX: NETCONF Server Configuration Model";
      }


      // Features

      feature ssh-listen {
        description
         "The ssh-listen feature indicates that the NETCONF server can
          open a port to listen for incoming client connections.";
      }

      feature ssh-call-home {
        description
         "The ssh-call-home feature indicates that the NETCONF server can
          connect to a client.";
        reference
         "RFC XXXX: Reverse Secure Shell (Reverse SSH)";
      }

      feature tls-listen {
        description
         "The tls-listen feature indicates that the NETCONF server can
          open a port to listen for incoming client connections.";
      }

      feature tls-call-home {
        description
         "The tls-call-home feature indicates that the NETCONF server can
          connect to a client.";
```

```
}


// top-level container  (groupings below)
container netconf-server {
  description
    "Top-level container for NETCONF server configuration.";

  uses session-options-container;
  uses listen-container;
  uses call-home-container;
  uses ssh-container;
  uses tls-container;

}


grouping session-options-container {
  description
    "";
  container session-options {
    description
      "NETCONF session options, independent of transport
       or connection strategy.";
    leaf hello-timeout {
      type uint32 {
        range "0 | 10 .. 3600";
      }
      units "seconds";
      default '600';
      description
        "Specifies the number of seconds that a session
         may exist before the hello PDU is received.
         A session will be dropped if no hello PDU
         is received before this number of seconds elapses.

         If this parameter is set to zero, then the server
         will wait forever for a hello message, and not
         drop any sessions stuck in 'hello-wait' state.

         Setting this parameter to zero may permit
         denial of service attacks, since only a limited
         number of concurrent sessions are supported
         by the server.";
    }
    leaf idle-timeout {
      type uint32 {
```

```
            range "0 | 10 .. 360000";
          }
          units "seconds";
          default '3600';
          description
            "Specifies the number of seconds that a session
             may remain idle without issuing any RPC requests.
             A session will be dropped if it is idle for an
             interval longer than this number of seconds.

             Sessions that have a notification subscription
             active are never dropped.

             If this parameter is set to zero, then the server
             will never drop a session because it is idle.";
        }
      }
    }

    grouping listen-container {
      description
        "";
      container listen {
        description
          "Configures listen behavior";
        //if-feature "(ssh-listen or tls-listen)";
        leaf max-sessions {
          type uint16 {
            range "0 .. 1024";
          }
          default '0';
          description
            "Specifies the maximum number of concurrent sessions
               that can be active at one time.  The value 0 indicates
               that no artificial session limit should be used.";
        }
        list endpoint {
          key name;
          description
            "List of endpoints to listen for connections on.";
          leaf name {
            type string;
            description
              "An arbitrary name for the listen endpoint.";
          }
          choice transport {
            mandatory true;
            description
```

```
            "Selects between SSH and TLS transports.";
          case ssh {
            if-feature ssh-listen;
            container ssh {
              description
                "SSH-specific listening configuration for inbound
                 connections.";
              uses address-and-port-grouping {
                refine port {
                  default 830;
                }
              }
              uses host-keys-container;
            }
          }
          case tls {
            if-feature tls-listen;
            container tls {
              description
                "TLS-specific listening configuration for inbound
                 connections.";
              uses address-and-port-grouping {
                refine port {
                  default 6513;
                }
              }
              uses certificates-container;
            }
          }
        }
        uses keep-alives-container  {
          refine keep-alives/interval-secs {
            default 0; // disabled by default for listen connections
          }
        }
      }
    }
  }

  grouping call-home-container {
    description
      "";
    container call-home {
      //if-feature "(ssh-call-home or tls-call-home)";
      description
        "Configures call-home behavior";
      list application {
        key name;
```

```
            description
              "List of applications to call-home to.";
            leaf name {
              type string;
              description
                "An arbitrary name for the remote application.";
            }
            choice transport {
              mandatory true;
              description
                "Selects between SSH and TLS transports.";
              case ssh {
                if-feature ssh-call-home;
                container ssh {
                  description
                    "Specifies SSH-specific call-home transport
                     configuration.";
                  uses endpoints-container {
                    refine endpoints/endpoint/port {
                      default 8888;  // pending IANA assignment
                    }
                  }
                  uses host-keys-container;
                }
              }
              case tls {
                if-feature tls-call-home;
                container tls {
                  description
                    "Specifies TLS-specific call-home transport
                     configuration.";
                  uses endpoints-container {
                    refine endpoints/endpoint/port {
                      default 9999;  // pending IANA assignment
                    }
                  }
                  uses certificates-container;
                }
              }
            }
            container connection-type {
              description
               "Indicates the NETCONF client's preference for how the
                device's connection is maintained.";
              choice connection-type {
                default persistent-connection;
                description
                  "Selects between persistent and periodic connections.";
```

```
                case persistent-connection {
                  container persistent {
                    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.";
                    uses keep-alives-container  {
                      refine keep-alives/interval-secs {
                        default 15; // 15 seconds for call-home sessions
                      }
                    }
                  }
                }
                case periodic-connection {
                  container periodic {
                    description
                     "Periodically connect to NETCONF client, using the
                      reconnection strategy, so the NETCONF client can
                      deliver pending messages to the NETCONF server.

                      For messages the NETCONF server wants to send to
                      to the NETCONF client, the NETCONF server should
                      proactively connect to the NETCONF client, if
                      not already, to send the messages immediately.";
                    leaf timeout-mins {
                      type uint8;
                      units minutes;
                      default 5;
                      description
                       "The maximum amount of unconnected time the
                        device will wait until establishing a
                        connection to the NETCONF client again. The
                        device MAY establish a connection before this
                        time if it has data it needs to send to the
                        NETCONF client. Note: this value differs from
                        the reconnection strategy's interval-secs
                        value.";
                    }
                    leaf linger-secs {
                      type uint8;
                      units seconds;
                      default 30;
```

```
                    description
                     "The amount of time the device should wait after
                      last receiving data from or sending data to the
                      NETCONF client's endpoint before closing its
                      connection to it.  This is an optimization to
                      prevent unnecessary connections.";
                  }
                }
              }
            }
            container reconnect-strategy {
              description
               "The reconnection strategy guides how a device reconnects
                to an application, after losing a connection to it,
                even if due to a reboot.  The device starts with the
                specified endpoint, tries to connect to it count-max
                times, waiting interval-secs between each connection
                attempt, 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.  NETCONF servers
                       SHOULD support this flag across reboots.";
                  }
                }
                default first-listed;
                description
                 "Specifies which of the application's endpoints the
                  device should start with when trying to connect to
                  the application.  If no previous connection has
                  ever been established, last-connected defaults to
                  the first endpoint listed.";
              }
              leaf interval-secs {
                type uint8;
                units seconds;
                default 5;
                description
                 "Specifies the time delay between connection attempts
```

```
                to the same endpoint.  Note: this value differs from
                the periodic-connection's timeout-mins value.";
            }
            leaf count-max {
              type uint8;
              default 3;
              description
               "Specifies the number times the device tries to
                connect to a specific endpoint before moving on to
                the next endpoint in the list (round robin).";
            }
          }
        }
      }
    }


    grouping ssh-container {
      description
        "";
      container ssh {
        description
          "Configures SSH properties not specific to the listen
           or call-home use-cases";
        //if-feature "(ssh-listen or ssh-call-home)";
        container host-keys {
          config false;
          description
            "Parent container for a list of host keys";
          list host-key {
            key name;
            description
              "A read-only list of host-keys supported by server";
            leaf name {
              type string;
              description
                "Common name for the host-key";
            }
            leaf format-identifier {
              type string;
              mandatory true;
              description
                "ssh-dss, ssh-rsa, x509v3-rsa2048-sha256, etc.";
              reference
                "RFC 4253: SSH Transport Layer Protocol, section 6.6
                 RFC 6187: X.509v3 Certificates for SSH, section 3";
            }
            leaf data {
```

```
              type binary;
              mandatory true;
              description
                "Key-specific binary encoding.";
              reference
                "RFC 4253: SSH Transport Layer Protocol, section 6.6";
            }
            leaf fingerprint {
              type string;
              mandatory true;
              description
                "c1:b1:30:29:d7:b8:de:6c:97:77:10:d7:46:41:63:87";
              reference
                "RFC 4716: The Secure Shell (SSH) Public Key File
                           Format, section 4";
            }
          }
        }
      }
    }


    grouping tls-container {
      description
        "";
      container tls {
        description
          "Configures TLS properties not specific to the listen
           or call-home use-cases";
        //if-feature "(tls-listen or tls-call-home)";
        container certificates {
          config false;
          description
            "Parent container for a list of certificates";
          list certificate {
            key name;
            description
              "A list of certificates";
            leaf name {
              type string;
              description
                "the certificate's common name";
            }
            leaf data {
              type binary;
              mandatory true;
              description
                "The binary certificate structure, as specified
```

```
                by RFC 5246, Section 7.4.2, i.e.,: opaque
                ASN.1Cert<1..2^24-1>;";
            }
          }
        }
        container client-auth {
          description
            "Container for TLS client authentication configuration.";
          container trusted-ca-certs {
            description
              "A list of Certificate Authority (CA) certificates that
               a NETCONF server can use to authenticate NETCONF client
               certificates.  A client's certificate is authenticated
               if there is a chain of trust to a configured trusted CA
               certificate.  Note, in the TLS protocol, the client
               certificate MAY be accompanied with additional
               certificates forming a chain of trust.  The client's
               certificate is authenticated if there is path-validation
               from any of the certificates it presents to a configured
               trust anchor.";
            leaf-list trusted-ca-cert {
              type binary;
              ordered-by system;
              description
                "The binary certificate structure as specified by RFC
                 5246, Section 7.4.6, i.e.,: opaque ASN.1Cert<1..2^24>;
                ";
              reference
                "RFC 5246: The Transport Layer Security (TLS)
                           Protocol Version 1.2";
            }
          }
          container trusted-client-certs {
            description
              "A list of client certificates that a NETCONF server can
               use to authenticate a NETCONF client's certificate.  A
               client's certificate is authenticated if it is an exact
               match to a configured trusted client certificates.";
            leaf-list trusted-client-cert {
              type binary;
              ordered-by system;
              description
                "The binary certificate structure, as
                 specified by RFC 5246, Section 7.4.6, i.e.,:

                   opaque ASN.1Cert<1..2^24>;

                ";
```

```
            reference
              "RFC 5246: The Transport Layer Security (TLS)
                         Protocol Version 1.2";
          }
        }
        container cert-maps {
          uses x509c2n:cert-to-name;
          description
           "The cert-maps container is used by a 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.";
        }
      }
    }
  }


  grouping host-keys-container {
    description
      "";
    container host-keys {
      description
        "Parent container for the list of host-keys.";
      leaf-list host-key {
        type string;
        min-elements 1;
        ordered-by user;
        description
          "User-ordered list of host-keys the SSH server
           considers when composing the list of  server
           host key algorithms it will send to the client.
           The value of the string is the name of a
           host-key configured on the system, as returned
           by /netconf-server/ssh/host-keys/host-key/name.";
        reference
          "RFC 4253: The SSH Transport Layer Protocol, Section 7";
      }
    }
  }

  grouping certificates-container {
    description
      "";
    container certificates {
```

```
        description
          "Parent container for the list of certificates.";
        leaf-list certificate {
          type string;
          min-elements 1;
          description
            "Unordered list of certificates the TLS server can
             pick from when sending its Server Certificate
             message.  The value of the string is the name of a
             certificate configured on the system, as returned by
             /netconf-server/tls/certificates/certificate/name";
          reference
            "RFC 5246: The TLS Protocol, Section 7.4.2";
        }
      }
    }

    grouping address-and-port-grouping {
      description
        "a common grouping";
      leaf address {
        type inet:ip-address;
        description
         "The IP address of the interface to listen on.";
      }
      leaf port {
        type inet:port-number;
        description
         "The local port number on this interface the
          NETCONF server listens on.";
      }
    }


    grouping endpoints-container {
      description
        "Grouping for transport-specific configuration for
         call-home connections.";
      container endpoints {
        description
          "Container for the list of endpoints.";
        list endpoint {
          key name;
          min-elements 1;
          ordered-by user;
          description
            "User-ordered list of endpoints for this application.
             Defining more than one enables high-availability.";
```

```
          leaf name {
            type string;
            description
              "An arbitrary name for the endpoint to connect to.";
          }
          leaf address {
            type inet:host;
            mandatory true;
            description
             "The hostname or IP address or hostname of the
              endpoint.  If a hostname is provided and DNS
              resolves to more than one IP address, the device
              SHOULD try all of the ones it can based on how
              its networking stack is configured (e.g. v4, v6,
              dual-stack).";
          }
          leaf port {
            type inet:port-number;
            description
             "The IP port for this endpoint. The device will use
              the IANA-assigned well-known port if not specified.";
          }
        }
      }
    }

    grouping keep-alives-container {
      description
        "";
      container keep-alives {
        description
          "Configures the keep-alive policy, to proactively
           test the aliveness of the NETCONF client, in
           order to know when a new call home connection
           should be established.  Keepalive implementation
           is described in RFC XXXX, section 4.";
        reference
          "RFC XXXX: NETCONF Server Configuration Model
           Section 4";
        leaf interval-secs {
          type uint8;
          units seconds;
          description
           "Sets a timeout interval in seconds after which
            if no data has been received from the NETCONF
            client, a message will be sent to request a
            response from the NETCONF client.  A value of
            '0' indicates that no keep-alive messages
```

```
            should be sent.";
        }
        leaf count-max {
          type uint8;
          default 3;
          description
           "Sets the number of keep-alive messages that
            may be sent without receiving any data from
            the NETCONF client before assuming the NETCONF
            client is no longer alive.  If this threshold
            is reached, the transport-level connection
            will be disconnected, which will trigger the
            reconnection strategy).  The interval timer is
            reset after each transmission, thus an
            unresponsive NETCONF client will be dropped
            after ~count-max * interval-secs seconds.";
        }
      }
    }
  }
```

<CODE ENDS>

## 4. Implementation strategy for keep-alives

One of the objectives listed above, Keep-alives for persistent connections (Section 2.6.6), indicates a need for a "keep-alive" mechanism. This section specifies how the NETCONF keep-alive mechanism is to be implemented for both the SSH and TLS transports.

Both SSH and TLS have the ability to support keep-alives securely. Using the strategies listed below, the keep-alive messages are sent inside the encrypted transport sessions.

### 4.1. Keep-alives for SSH

The SSH keep-alive solution that is expected to be used is ubiquitous in practice, though never being explicitly defined in an RFC. The strategy used is to purposely send a malformed request message with a flag set to ensure a response. More specifically, per section 4 of [RFC4253], either SSH peer can send a SSH_MSG_GLOBAL_REQUEST message with "want reply" set to '1' and that, if there is an error, will get back a SSH_MSG_REQUEST_FAILURE response. Similarly, section 5 of [RFC4253] says that either SSH peer can send a SSH_MSG_CHANNEL_REQUEST message with "want reply" set to '1' and that, if there is an error, will get back a SSH_MSG_CHANNEL_FAILURE response.

To ensure that the request will fail, current implementations of this keep-alive strategy (e.g. OpenSSH's `sshd` server) send an invalid "request name" or "request type", respectively. Abiding to the extensibility guidelines specified in Section 6 of [RFC4251], these implementations use the "name@domain". For instance, when configured to send keep-alives, OpenSSH sends the string "keepalive@openssh.com". In order to remain compatible with existing implementations, this draft does not require a specific "request name" or "request type" string be used, implementations are free to pick values of their choosing.

### 4.2. Keep-alives for TLS

The TLS keep-alive solution that is expected to be used is defined in [RFC6520]. This solution allows both peers to advertise if they can receive heartbeat request messages from its peer. For standard NETCONF over TLS connections, devices SHOULD advertise "peer_allowed_to_send", as per [RFC6520]. This advertisement is not a "MUST" in order to grandfather existing NETCONF over TLS implementations. For NETCONF Call Home, the network management system MUST advertise "peer_allowed_to_send" per [RFC6520]. This is a "MUST" so as to ensure devices can depend in it always being there for call home connections, which is when keep-alives are needed the most.

## 5. Security Considerations

The YANG modules defined in this memo are designed to be accessed via the NETCONF protocol [RFC6241]. Authorization for access to specific portions of conceptual data and operations within this module is provided by the NETCONF access control model (NACM) [RFC6536].

There are a number of data nodes defined in the "ietf-netconf-server" YANG module which are readable and/or writable that may be considered sensitive or vulnerable in some network environments. Write and read operations to these data nodes can have a negative effect on network operations. It is thus important to control write and read access to these data nodes. Below are the data nodes and their sensitivity/ vulnerability.

netconf-server/tls/client-auth/trusted-ca-certs:

- This container contains certificates that the system is to use as trust anchors for authenticating TLS-specific client certificates. Write access to this node should be protected.

netconf-server/tls/client-auth/trusted-client-certs:

- This container contains certificates that the system is to trust directly when authenticating TLS-specific client certificates. Write access to this node should be protected.

netconf-server/tls/client-auth/cert-map:

- This container contains a user name that some deployments may consider sensitive information. Read access to this node may need to be guarded.

## 6. IANA Considerations

This document registers two URIs in the IETF XML registry [RFC2119]. Following the format in [RFC3688], the following registrations are requested:

```
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.

URI: urn:ietf:params:xml:ns:yang:ietf-system-tle-auth
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

This document registers two YANG modules in the YANG Module Names registry [RFC6020].

```
name:         ietf-netconf-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-netconf-server
prefix:       ncserver
reference:    RFC XXXX

name:         ietf-system-tls-auth
namespace:    urn:ietf:params:xml:ns:yang:ietf-system-tls-auth
prefix:       sys-tls-auth
reference:    RFC XXXX
```

## 7. Other Considerations

The YANG module define herein does not itself support virtual routing and forwarding (VRF). It is expected that external modules will augment in VRF designations when needed.

## 8. 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, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, and Phil Shafer.

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

## 9. References

### 9.1. Normative References

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

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

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

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, February 2012.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

[draft-ietf-netconf-call-home] Watsen, K., "NETCONF Call Home", draft-ieft-netconf-call-home-00 (work in progress), 2014.

[draft-ietf-netmod-snmp-cfg]
Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", draft-ietf-netmod-snmp-cfg-08 (work in progress), September 2014.

[rfc5539bis]
Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS)", draft-ietf-netconf-rfc5539bis-04 (work in progress), October 2013.

## 9.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

# Appendix A. Examples

## A.1. SSH Transport Configuration + State

The following example illustrastes the <get> response from a NETCONF server that only supports SSH, both listening for incoming connections as well as calling home to a single application having two endpoints. Please also note that the list of host-keys at the end is read-only operational state.

```
<netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <listen>
    <endpoint>
      <name>foo bar</name>
      <ssh>
        <address>11.22.33.44</address>
        <host-keys>
          <host-key>my-rsa-key</host-key>
          <host-key>my-dss-key</host-key>
        </host-keys>
      </ssh>
    </endpoint>
  </listen>
  <call-home>
    <application>
      <name>config-mgr</name>
      <ssh>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <address>11.22.33.44</address>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <address>55.66.77.88</address>
          </endpoint>
        </endpoints>
        <host-keys>
          <host-key>my-call-home-x509-key</host-key>
        </host-keys>
      </ssh>
    </application>
  </call-home>
  <ssh>
    <host-keys>
      <host-key>
        <name>my-rsa-key</name>
        <format-identifier>ssh-rsa</format-identifier>
```

```
        <data>  <!-- base64 reformated for draft -->
          AAAAB3NzaC1yc2EAAAABIwAAAQEA7D2lxYg3+WD97RZqZtO8bUU8QpIl6g9
          X11kZHZ8NgSIR+x2H1MHCD5sEjmx/B6JIouK5eBvbJE9FFV3phsl62fupN6
          Y4EmXosC6iqpuI41dcGA63XCQ1OenWG4ppdq1f8tlecSrmEcLw7MKPzBHK6
          rNQTciqMuVuLPOKwBu/54QAiUwvvHKAsk8bkN9YxEJ1NTV1FFQmvMOADVcD
          2qqPangETwV5zInW8AEkBbLccM/mmHucGNS81axXR3V9R5KgXF2DyGB47d2
          k6iOnGa3LBIOYi/5Q+O8IFUlO+kytfqwuFgUc+Mx7aKReSIAPov3owVjeBL
          KWsvjD24UO68qtwQ==
        </data>
        <fingerprint>
          c1:b1:30:29:d7:b8:de:6c:97:77:10:d7:46:41:63:87
        </fingerprint>
      </host-key>
      <host-key>
        <name>my-dss-key</name>
        <format-identifier>ssh-dss</format-identifier>
        <data>  <!-- base64 reformated for draft -->
          AAAAB3NzaC1kc3MAAACBAIq7XfGmZKJgibJEIMzj70YMVfpeewBCj89VrUS
          gLsJmxP/TrXFuhzW2UIaI8sePMYUXj/Vgp5DUD+eBSBkHMH4ga0U5t/clqn
          y73x8Vg6LQg9f0OTaUnpRWbWrdac7U5/BRBTtMA3amHZhHrKs7BrCepS/y8
          cUbxBCPF3aYMK/5AAAAFQC7wetEbDwghYtz8Z3xIwDdxs6mOwAAAIBursEk
          jnvs5zzyUH7iNiyBojDoyrsq81jPM6KopkfA5Ypp2KTySPev/mkL0SoVfIb
          +HttVfQ3Q63+sf1Qyk+gUtniSdN2AqtFQYKxtTcXim4McWk6IixkYFP8kkt
          02t9Hsl0eXvltmogrlRsiuJsTAbFS+QTeq4OGTODCT5jjVdQAAAIA2llpZg
          y5v46lGt4dQhkH8ytyMGyjBRPF6rm51msinX3lMR9xfwTaS7ZYP0b6HJt5M
          sQI+m7iIYaVFB1oC8niXbkkavLcxhGpNVkwE2INWS4TIBbTQhivuoE+dMYY
          KauLQxqSUjixJk3LjhCQb
        </data>
        <fingerprint>
          c1:b1:30:29:d7:b8:de:6c:97:77:10:d7:46:41:63:87
        </fingerprint>
      </host-key>
      <host-key>
        <name>my-call-home-x509-key</name>
        <format-identifier>x509v3-rsa2048-sha256</format-identifier>
        <data>    <!-- base64 reformated for draft -->
          AAAAB3NzaC1yc2EAAAABIwAAAQEAyBLl90dPUGX7Es12q7YKkw6v8WgWop+
          B62zhT39C+yvslMIwIqgHYii0h/TGktahKpBwssawfhvAZoMF/nOyO3yDPD
          pQxNrA76H7owNOjG5206QHDYfVALKPvxgrDy/6BjsR9MayOGkZTSL6GRFSl
          g7ivT9AIR9E5qXmP+1z+IDufRlpwfaGfpZAxjJLEwzAjFAIwXsXKJ5FH/QP
          mfC6gxfhqpt9rJCDlgqmzrXi8dXKsFUC3/o1lzezqTXTV1iMETTuCHgWegF
          5QcX2baBdFgCnkd1SnftVoBHVnvXA1euRqgiG3fMNK4rct0D99D+GI+kZc+
          vQyUdCw3dPlhXPZw==
        </data>
        <fingerprint>
          97:77:10:29:d7:b8:de:6c:97:77:30:29:d7:41:63:87
        </fingerprint>
      </host-key>
```

```
    </host-keys>
  </ssh>
</netconf-server>
```

## A.2. TLS Transport Configuration + State

The following example illustrastes the <get> response from a NETCONF server that only supports TLS, both listening for incoming connections as well as calling home to a single application having two endpoints. Please note also the configurations for authenticating client certificates and mappings authenticated certificates to NETCONF user names.

```
<netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <listen>
    <endpoint>
      <name>primary-netconf-endpoint</name>
      <tls>
        <address>11.22.33.44</address>
        <certificates>
          <certificate>fw1.east.example.com</certificate>
        </certificates>
      </tls>
    </endpoint>
  </listen>
  <call-home>
    <application>
      <name>config-mgr</name>
      <tls>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <address>11.22.33.44</address>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <address>55.66.77.88</address>
          </endpoint>
        </endpoints>
        <certificates>
          <certificate>fw1.east.example.com</certificate>
        </certificates>
      </tls>
    </application>
  </call-home>
  <tls>
    <certificates>
      <certificate>
```

```
        <name>fw1.east.example.com</name>
        <data>  <!-- base64 reformated for draft -->
          AAAAB3NzaC1yc2EAAAABIwAAAQEA7D2lxYg3+WD97RZqZtO8bUU8QpIl6g9
          X11kZHZ8NgSIR+x2H1MHCD5sEjmx/B6JIouK5eBvbJE9FFV3phsl62fupN6
          Y4EmXosC6iqpuI41dcGA63XCQ1OenWG4ppdq1f8tlecSrmEcLw7MKPzBHK6
          rNQTciqMuVuLPOKwBu/54QAiUwvvHKAsk8bkN9YxEJ1NTV1FFQmvMOADVcD
          2qqPangETwV5zInW8AEkBbLccM/mmHucGNS81axXR3V9R5KgXF2DyGB47d2
          k6iOnGa3LBIOYi/5Q+O8IFUlO+kytfqwuFgUc+Mx7aKReSIAPov3owVjeBL
          KWsvjD24UO68qtwQ==
        </data>
      </certificate>
    </certificates>
    <client-auth>
      <trusted-ca-certs>
        <trusted-ca-cert>
          QW4gRWFzdGVyIGVnZywgZm9yIHRob3NlIHdobyBtaWdodCBsb29rICA6KQo=
        </trusted-ca-cert>
      </trusted-ca-certs>
      <trusted-client-certs>
        <trusted-client-cert>
          SSBhbSB0aGUgZWdnIG1hbiwgdGhleSBhcmUgdGhlIGVnZyBtZW4uCg==
        </trusted-client-cert>
        <trusted-client-cert>
          SSBhbSB0aGUgd2FscnVzLCBnb28gZ29vIGcnam9vYi4K
        </trusted-client-cert>
      </trusted-client-certs>
      <cert-maps>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>Joe Cool</name>
        </cert-to-name>
      </cert-maps>
    </client-auth>
  </tls>
</netconf-server>
```

## Appendix B. Change Log

### B.1. 00 to 01

- Restructured document so it flows better
- Added trusted-ca-certs and trusted-client-certs objects into the ietf-system-tls-auth module

### B.2. 01 to 02

- removed the "one-to-many" construct
- removed "address" as a key field
- removed "network-manager" terminology
- moved open issues to github issues
- brought TLS client auth back into model

### B.3. 02 to 03

- fixed tree diagrams and surrounding text

### B.4. 03 to 04

- reduced the number of grouping statements
- removed psk-maps and associated feature statements
- added ability for listen/call-home instances to specify which host-keys/certificates (of all listed) to use
- clarified that last-connected should span reboots
- added missing "objectives" for selecting which keys to use, authenticating client-certificates, and mapping authenticated client-certificates to usernames
- clarified indirect client certificate authentication
- added keep-alive configuration for listen connections
- added global-level NETCONF session parameters

## Appendix C. Open Issues

Please see: https://github.com/netconf-wg/server-model/issues.

## Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Juergen Schoenwaelder
Jacobs University Bremen

EMail: j.schoenwaelder@jacobs-university.de