NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

K. Watsen
Juniper Networks
J. Schoenwaelder
Jacobs University Bremen

February 14, 2014

A YANG Data Model for NETCONF Server Configuration
draft-kwatsen-netconf-server-01

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 they support device-initiated connections, 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 August 18, 2014.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Terminology 3
1.2. Tree Diagrams 3
2. Objectives 3
2.1. Support all NETCONF Transports 3
2.2. Align Transport-Specific Configurations 4
2.3. Support Transport-Independent Configuration 4
2.4. Support both Inbound and Outbound Connections 4
2.5. For Device-Initiated Outbound Connections 4
2.5.1. Support More than One Application 4
2.5.2. Support Applications Having More than One Server . . 4
2.5.3. Support a Reconnection Strategy 5
2.5.4. Support both Persistent and Periodic Connections . . 5
2.5.5. Keep-Alives for Persistent Connections 5
2.5.6. Customizations for Periodic Connections 5
3. Data Model Overview 6
3.1. The "listen" Grouping 6
3.2. The "call-home" Grouping 6
3.3. The SSH Subtree 7
3.4. The TLS Subtree 8
4. NETCONF Server YANG Module 9
5. Security Considerations 21
6. IANA Considerations 21
7. Acknowledgements 21
8. References 21
8.1. Normative References 22
8.2. Informative References 22
Appendix A. Example: SSH 22
Appendix B. Example: TLS 23
Appendix C. Change Log 25
C.1. 00 to 01 25
Appendix D. Open Issues 25

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 it supports, what ports they listen on, whether they support device-initiated connections, 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 the data model 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 this YANG module is to enable the configuration of the NETCONF service on the device. This scope includes both transport-independent and transport-specific configuration parameters.

2.1. Support all NETCONF Transports

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

Since implementations may not support all transports, the module should use YANG "feature" statements so that each implementation can advertise which transports it actually supports.

2.2. Align Transport-Specific Configurations

While each transport is unique in its protocol and may have some distinct configurations, there remains a significant overlap between them. Thus the YANG module should use "grouping" statements so that the common aspects can be configured similarly.

2.3. Support Transport-Independent Configuration

Since some NETCONF server configurations may be independent of any transport, the module should define a location for these transport-independent values to be configured.

2.4. Support both Inbound and Outbound Connections

Historically, NETCONF only supported the device opening a port to listen for inbound client connections. However, the NETCONF working group is actively defining support for devices to initiate outbound connections (e.g., "call home"). Thus, the module should enable the configuration of both inbound and outbound connections.

Since implementations may not support both inbound and outbound connections, the module should use YANG "feature" statements so that each implementation can advertise the type of connections it actually supports.

2.5. For Device-Initiated Outbound Connections

The following objectives only pertain to support for device-initiated outbound connections.

2.5.1. Support More than One Application

A device may be managed by more than one northbound applications. 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 management connections, it should be able to do so for more than one application.

2.5.2. 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 connections to the application, it should be able to connect to any of the applications servers.

2.5.3. Support a Reconnection Strategy

Assuming an application has more than one server, then it becomes necessary to understand how a device should reconnect to the application should it lose its connection to one of 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.

Note that 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 the device how to get connected to another server.

2.5.4. 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.

2.5.5. Keep-Alives for Persistent Connections

If a persistent connection is desired, it is the responsibility of the connection-initiator to actively test the connection for aliveness. However, there is a balance between the frequency of the tests and the networking overhead they generate. The appropriate balance can only be determined by the application, based on its interaction requirements. Therefore, for persistent connections, keep-alive settings should be configurable on a per-application basis.

2.5.6. 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. Note, 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 Overview

3.1. The "listen" Grouping

To enable transports to configure listening on one or more ports in a common way, this grouping is defined. Being a grouping enables each transport-specific data-model to augment it as needed (e.g., to specify a default for the "port" values), as well as enable implementations to advertise support for listening for inbound connections using a YANG feature.

```
+--rw listen
   +--rw (one-or-many)?
      +--:(one-port)
      |  +--rw port?         inet:port-number
      +--:(many-ports)
         +--rw interface* [address]
            +--rw address    inet:ip-address
            +--rw port?      inet:port-number
```

3.2. The "call-home" Grouping

To enable transports to configure initiating connections to remote applications in a common way, this grouping is defined. Being a grouping enables each transport-specific data-model to augment it as needed (e.g., to specify a default port value, lists of algorithms to advertise, etc.), as well as enable implementations to advertise support for listening for inbound connections using a YANG feature.

```
+--rw call-home
   +--rw applications
      +--rw application* [name]
         +--rw name                  string
         +--rw description?          string
         +--rw servers
         |  +--rw server* [address]
```

```
            |     +--rw address    inet:host
            |     +--rw port?      inet:port-number
            +--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
```

3.3. The SSH Subtree

The SSH subtree uses both the "listen" and "call-home" groupings mentioned above. Support for the SSH transport is advertised by the "ssh" feature, while listening for clients and calling home are advertised by the "inbound-ssh" and "outbound-ssh" features respectively. The SSH subtree augments the "call-home" grouping by adding a "host-keys" container. Also, though not visible in the tree output below, this subtree refines all the port values with a suitable default (i.e., 830).

```
+--rw netconf
   +--rw ssh {ssh}?
      +--rw listen {inbound-ssh}?
      |  +--rw (one-or-many)?
      |     +--:(one-port)
      |     |  +--rw port?        inet:port-number
      |     +--:(many-ports)
      |        +--rw interface* [address]
      |           +--rw address    inet:ip-address
      |           +--rw port?      inet:port-number
      +--rw call-home {outbound-ssh}?
         +--rw applications
            +--rw application* [name]
               +--rw name                  string
               +--rw description?          string
               +--rw servers
               |  +--rw server* [address]
               |     +--rw address    inet:host
               |     +--rw port?      inet:port-number
```

```
                +--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
                +--rw host-keys
                   +--rw host-key* [name]
                      +--rw name    string
```

3.4. The TLS Subtree

The TLS subtree uses both the "listen" and "call-home" groupings mentioned above, while also defining containers for certificate and pre-shared key mappings. Support for the TLS transport is advertised by the "tls" feature, while listening for clients and calling home are advertised by the "inbound-tls" and "outbound-tls" features respectively. Also, though not visible in the tree output below, this submodule refines all the port values with a suitable defaults (e.g., 6513).

```
+--rw netconf
   +--rw tls {tls}?
      +--rw listen {inbound-tls}?
      |  +--rw (one-or-many)?
      |     +--:(one-port)
      |     |  +--rw port?        inet:port-number
      |     +--:(many-ports)
      |        +--rw interface* [address]
      |           +--rw address    inet:ip-address
      |           +--rw port?      inet:port-number
      +--rw call-home {outbound-tls}?
      |  +--rw applications
      |     +--rw application* [name]
      |        +--rw name                  string
      |        +--rw description?          string
      |        +--rw servers
      |        |  +--rw server* [address]
      |        |     +--rw address    inet:host
```

```
          |        |     +--rw port?      inet:port-number
          |        +--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
          +--rw cert-maps {tls-map-certificates}?
          |  +--rw cert-to-name* [id]
          |     +--rw id             uint32
          |     +--rw fingerprint    x509c2n:tls-fingerprint
          |     +--rw map-type       identityref
          |     +--rw name           string
          +--rw psk-maps {tls-map-pre-shared-keys}?
             +--rw psk-map* [psk-identity]
                +--rw psk-identity        string
                +--rw user-name           nacm:user-name-type
                +--rw not-valid-before?   yang:date-and-time
                +--rw not-valid-after?    yang:date-and-time
                +--rw key                 yang:hex-string
```

4. NETCONF Server YANG Module

This YANG module imports YANG extensions from [RFC6536], and imports YANG types from [RFC6991] and a YANG grouping from [I-D.ietf-netmod-snmp-cfg].

```
module ietf-netconf-server {

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

  import ietf-yang-types {
    prefix yang;               // RFC 6991
  }
  import ietf-inet-types {
    prefix inet;               // RFC 6991
  }
  import ietf-x509-cert-to-name {
```

```
      prefix x509c2n;             // I-D.ietf-netconf-rfc5539bis
    }
    import ietf-netconf-acm {
      prefix nacm;                // RFC 6536
    }

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

                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 "2014-01-24" {
      description
```

```
        "Initial version";
      reference
        "RFC XXXX: A YANG Data Model for NETCONF Server Configuration";
    }

    /*
     * Features
     */

    feature ssh {
      description
       "A server implements this feature if it supports NETCONF
        over Secure Shell (SSH).";
      reference
       "RFC 6242: Using the NETCONF Protocol over Secure Shell (SSH)";
    }

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

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

    feature tls {
      description
       "A server implements this feature if it supports NETCONF
        over Transport Layer Security (TLS).";
      reference
       "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
    }

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

    feature outbound-tls {
      description
       "The outbound-tls feature indicates that the server can
```

```
         connect to a client.";
      }

      feature tls-map-certificates {
        description
         "The tls-map-certificates feature indicates that the
          server implements mapping X.509 certificates to NETCONF
          usernames.";
      }

      feature tls-map-pre-shared-keys {
        description
         "The tls-map-pre-shared-keys feature indicates that the
          server implements mapping TLS pre-shared keys to NETCONF
          usernames.";
      }

      /*
       * Groupings
       */

       grouping listen-config {
         description
          "Provides a choice of configuring one of more ports
           to listen for incoming client connections.";

         choice one-or-many {
           default one-port;
           case one-port {
             leaf port {
               type inet:port-number;
               description
                "The port number the server listens on on all
                 interfaces.";
             }
           }

           case many-ports {
             list interface {
               key "address";
               leaf address {
                 type inet:ip-address;
                   mandatory true;
                   description
                    "The local IP address of the interface to listen
                     on.";
                 }
                 leaf port {
```

```
                  type inet:port-number;
                  description
                   "The local port number on this interface the
                    server listens on.";
                }
              }
            }
          }
        }

    grouping call-home-config {

      container applications {
        description
         "A list of applications the device iniates connections
          to. The configuration for each application specifies
          its details, including its servers, the type of
          connection to maintain, and the reconnection strategy
          to use.";

        list application {
          key name;
  //        min-elements 1;  // this forces <call-home>?!
          leaf name {
            type string {
              length 1..64;  // XXX why these limits?
            }
            mandatory true;
            description
             "An arbitrary name for the application the device
              is connecting to.";
          }
          leaf description {
            type string;
            description
              "An optional description for the application.";
          }
          container servers {
            description
             "An ordered listing of the application's servers
              that the device should attempt connecting to.";
            list server {
              key address;
              min-elements 1;
              ordered-by user;
              leaf address {
                type inet:host;
                mandatory true;
```

```
            description
             "The address or domain-name of the server.";
          }
          leaf port {
            type inet:port-number;
            description
             "The IP port for this server. The device will use
              the IANA-assigned well-known port if not specified.";
          }
        }
      }
      container connection-type {
        description
         "Indicates the application's preference for how the
          device's connection is maintained.";
        choice connection-type {
          default persistent-connection;

          case persistent-connection {
            container persistent {
              description
               "Maintain a persistent connection to the
                application. If the connection goes down,
                immediately start trying to reconnect to it,
                using the reconnection strategy.

                This connection type minimizes any
                application-to-server data-transfer delay,
                albeit at the expense of holding resources
                longer.";
              container keep-alives {
                leaf interval-secs {
                  type uint8;
                  units seconds;
                  default 15;
                  description
                   "Sets a timeout interval in seconds after which
                    if no data has been received from the
                    application, a message will be sent to request
                    a response from the application.  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
                    application before assuming the application is
                    no longer alive.  If this threshold is reached,
                    the transport-level connection will be
                    disconnected (thus triggering the reconnection
                    strategy).  The interval timer is reset after
                    each transmission, thus an unresponsive
                    application will be disconnected after
                    approximately count-max * interval-secs
                    seconds.";
                }
              }
            }
          }

          case periodic-connection {
            container periodic {
              description
               "Periodically connect to application, using the
                reconnection strategy, so it can flush any pending
                data it may be holding. This connection type
                minimizes resources held open, albeit at the
                expense of longer application-to-server
                data-transfer delay.  Note that for
                server-to-application data, the data should be
                sent immediately, connecting to application first
                if not already.";
              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 application again. The
                  device may establish a connection before this
                  time if it has data it needs to send to the
                  application. 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
```

```
                application before closing its connection to it.
                This is an optimization to prevent unnecessary
                connections.";
            }
          }
        }
      }
    }

    // XXX
    // Should we have something smarter as the reconnect
    // strategy, e.g. an exponential backoff?

    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
        server, tries to connect to it count-max times, waiting
        interval-secs between each connection attempt, before
        trying the next server in the list (round robin).";
      leaf start-with {
        type enumeration {
          enum first-listed { value 1; }
          enum last-connected { value 2; }
        }
        default first-listed;
        description
         "Specifies which of the application's servers the
          device should start with when trying to connect to
          the application. In the case of newly configured
          application, the first server listed shall be
          considered last-connected.";
      }
      leaf interval-secs {
        type uint8;
        units seconds;
        default 5;
        description
         "Specifies the time delay between connection attempts
          to the same server.  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 server before moving on to
                the next server in the list (round robin).";
            }
          }
        }
      }
    }


    grouping ssh-config {
       description
        "Provides a reusable grouping for all the ssh config.  This
         is done primarily to enable external modules to reference
         this definition in a &amp;quot;uses&amp;quot; statement.";

        container listen {
          if-feature inbound-ssh;
          description
           "Provides the configuration of the NETCONF server to
            open one or more ports to listen for incoming client
            connections.";
          uses listen-config {
            refine one-or-many/one-port/port {
              default 830;
            }
            refine one-or-many/many-ports/interface/port {
              default 830;
            }
          }
        }

        container call-home {
          if-feature outbound-ssh;
          description
           "Provides the configuration of the NETCONF call-home
            clients to connect to, the overall call-home policy,
            and the reconnect strategy.";
          uses call-home-config {
            augment applications/application {
              container host-keys {
                description
                 "An ordered listing of the SSH host keys the
                  device should advertise to the application.";
                list host-key {
                  key name;
                  min-elements 1;
                  ordered-by user;
                  leaf name {
```

```
                    type string {
                      length 1..64; // XXX why this limit?
                    }
                    mandatory true;
                    description
                     "The name of a host key the device should
                      advertise during the SSH key exchange.";
                  }
                }
              }
            }
          }
        }
      }

     grouping tls-config {
        description
         "Provides a reusable grouping for all the tls config.  This
          is done primarily to enable external modules to reference
          this definition in a &amp;quot;uses&amp;quot; statement.";

         container listen {
           if-feature inbound-tls;
           description
            "Provides the configuration of the NETCONF server to
             open one or more ports to listen for incoming client
             connections.";
           uses listen-config {
             refine one-or-many/one-port/port {
               default 6513;
             }
             refine one-or-many/many-ports/interface/port {
               default 6513;
             }
           }
         }

         container call-home {
           if-feature outbound-tls;
           description
            "Provides the configuration of the NETCONF call-home
             clients to connect to, the overall call-home policy,
             and the reconnect strategy.";
           uses call-home-config;
         }

         /*
          * Objects for deriving NETCONF usernames from X.509
```

```
         * certificates.
         */

        container cert-maps {
          if-feature tls-map-certificates;
          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.";
        }

        /*
         * Objects for deriving NETCONF usernames from TLS
         * pre-shared keys.
         */

        container psk-maps {
          if-feature tls-map-pre-shared-keys;
          description
           "During the TLS Handshake, the client indicates which
            key to use by including a PSK identity in the TLS
            ClientKeyExchange message. On the server side, this
            PSK identity is used to look up an entry in the psk-map
            list. If such an entry is found, and the pre-shared keys
            match, then the client is authenticated. The server uses
            the value from the user-name leaf in the psk-map list as
            the NETCONF username.  If the server cannot find an entry
            in the psk-map list, or if the pre-shared keys do not
            match, then the server terminates the connection.";
          reference
           "RFC 4279: Pre-Shared Key Ciphersuites for Transport Layer
                      Security (TLS)";

          list psk-map {
            key psk-identity;

            leaf psk-identity {
              type string;
              description
               "The PSK identity encoded as a UTF-8 string. For
                details how certain common PSK identity formats can
                be encoded in UTF-8, see section 5.1. of RFC 4279.";
              reference
```

```
              "RFC 4279: Pre-Shared Key Ciphersuites for Transport
                         Layer Security (TLS)";
            }
            leaf user-name {
              type nacm:user-name-type;
              mandatory true;
              description
               "The NETCONF username associated with this PSK
                identity.";
            }
            leaf not-valid-before {
              type yang:date-and-time;
              description
               "This PSK identity is not valid before the given date
                and time.";
            }
            leaf not-valid-after {
              type yang:date-and-time;
              description
               "This PSK identity is not valid after the given date
                and time.";
            }
            leaf key {
              type yang:hex-string;
              mandatory true;
              nacm:default-deny-all;
              description
                "The key associated with the PSK identity";
              reference
                "RFC 4279: Pre-Shared Key Ciphersuites for Transport
                 Layer Security (TLS)";
            }
          }
        }
      }


    /*
     * Configuration data nodes
     */

    container netconf {
      description
       "Top-level container for NETCONF server related
        configuration objects.";

      container ssh {
        if-feature ssh;
```

```
          uses ssh-config;
      }

      container tls {
        if-feature tls;
          uses tls-config;
      }

    }
  }
```

5. Security Considerations

This document defines a YANG modules to configure NETCONF's SSH and TLS transports. Please see the Security Considerations section in those RFCs for transport-specific issues.

6. IANA Considerations

This document registers one URIs in the IETF XML registry [RFC2119]. Following the format in [RFC3688], the following registration is requested to be made.

```
   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.
```

This document registers one YANG module 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
```

7. Acknowledgements

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

8. References

8.1. Normative References

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

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

[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.

[I-D.ietf-netmod-snmp-cfg] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", draft-ietf-netmod-snmp-cfg-03 (work in progress), November 2013.

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

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

[I-D.ietf-netconf-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.

8.2. Informative References

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

Appendix A. Example: SSH

```
<netconf xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <ssh>

    <listen>
      <port>831</port>
    </listen>
```

```
       <call-home>
         <applications>
           <application>
             <name>config-mgr</name>
             <description>
                This entry requests the device to periodically
                connect to the Configuration Manager application
             </description>
             <servers>
                <server>
                   <address>config-mgr1.example.com</address>
                </server>
                <server>
                   <address>config-mgr2.example.com</address>
                </server>
             </servers>
             <connection-type>
               <periodic>
                 <timeout-mins>5</timeout-mins>
                 <linger-secs>10</linger-secs>
               </periodic>
             </connection-type>
             <reconnect-strategy>
                <start-with>last-connected</start-with>
                <interval-secs>10</interval-secs>
                <count-max>3</count-max>
             </reconnect-strategy>
             <host-keys>
                <host-key>
                   <name>ssh_host_key_cert</name>
                </host-key>
                <host-key>
                   <name>ssh_host_key_cert2</name>
                </host-key>
             </host-keys>
           </application>
         </applications>
       </call-home>

     </ssh>
   </netconf>
```

Appendix B. Example: TLS

```
   <netconf xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
     <tls>

       <listen>
```

```
          <interface>
            <address>192.0.2.1</address>
            <port>6514</port>
          </interface>
        </listen>

        <call-home>
          <applications>
            <application>
              <name>log-monitor</name>
              <description>
                 This entry requests the device to maintain a
                 persistent connect to the Log Monitor application
              </description>
              <servers>
                 <server>
                    <address>log-monitor1.example.com</address>
                 </server>
                 <server>
                    <address>log-monitor2.example.com</address>
                 </server>
              </servers>
              <connection-type>
                <persistent>
                  <keep-alives>
                    <interval-secs>5</interval-secs>
                    <count-max>3</count-max>
                  </keep-alives>
                </persistent>
              </connection-type>
              <reconnect-strategy>
                 <start-with>first-listed</start-with>
                 <interval-secs>10</interval-secs>
                 <count-max>4</count-max>
              </reconnect-strategy>
            </application>
          </applications>
        </call-home>

        <cert-maps>
          <!-- Use a subject alt name field of a specific
               certificate as the NC username. -->
          <cert-to-name>
            <id>1</id>
            <fingerprint>11:0A:05:11:00</fingerprint>
            <map-type>x509c2n:san-any</map-type>
          </cert-to-name>
          <!-- Map a specific certificate to the NC username
```

```
             'Joe Cool'. -->
          <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>

        <psk-maps>
          <psk-map>
            <psk-identity>a8gc8]klh59</psk-identity>
            <user-name>admin</user-name>
            <not-valid-before>2013-01-01T00:00:00Z</not-valid-before>
            <not-valid-after>2014-01-01T00:00:00Z</not-valid-after>
          </psk-map>
        </psk-maps>

      </tls>
    </netconf>
```

Appendix C. Change Log

C.1. 00 to 01

- Restructured YANG module slightly, to provide groupings useful to the ZeroTouch draft.

Appendix D. Open Issues

- NETCONF implementations typically have config parameters such as session timeouts or hello timeouts. Shall they be included in this model?

- Do we need knobs to enable/disable call-home without the need to remove all the call-home client configuration?

- Do we need something equivalent to the host-keys in the TLS configuration subtree?

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Juergen Schoenwaelder
Jacobs University Bremen

EMail: j.schoenwaelder@jacobs-university.de