NETCONF Server Configuration Model
draft-ietf-netconf-server-model-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 December 03, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
   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 both Listening for Connections and Call Home .. 4
   2.4. For Call Home Connections ........................... 4
      2.4.1. Support More than One Application ............... 4
      2.4.2. Support Applications Having More than One Server .. 4
      2.4.3. Support a Reconnection Strategy ............... 4
      2.4.4. Support both Persistent and Periodic Connections .. 5
      2.4.5. Reconnection Strategy for Periodic Connections .. 5
      2.4.6. Keep-Alives for Persistent Connections .......... 5
      2.4.7. Customizations for Periodic Connections ....... 5
3. Data Model .................................................. 6
   3.1. Overview .............................................. 6
   3.2. YANG Module ........................................... 7
4. Keep-Alives for SSH and TLS ................................ 16
   4.1. SSH ................................................... 16
   4.2. TLS .................................................. 17
5. User Authentication for TLS .................................. 17
   5.1. Introduction .......................................... 17
   5.2. Data Model Overview .................................. 17
   5.3. YANG Module ........................................... 18
6. Security Considerations ....................................... 23
7. IANA Considerations ........................................... 24
8. Other Considerations .......................................... 24
9. Acknowledgements ............................................. 24
10. References .................................................. 24
    10.1. Normative References ................................ 24
    10.2. Informative References ............................. 26
Appendix A. Example: SSH Transport Configuration .............. 26
Appendix B. Example: TLS Transport Configuration ............. 27
Appendix C. Example: TLS Authentication Configuration ....... 28
Appendix D. Change Log ......................................... 29
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 does the server listen 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 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 [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 implementation can accurately advertise which transports are supported.

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 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 ([I-D.ietf-netconf-rfc5539bis] and [draft-ietf-netconf-reverse-ssh]). The module should configure both listening for connections and call-home.

Since 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.

2.4. For Call Home Connections

The following objectives only pertain to call home connections.

2.4.1. 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.

2.4.2. 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 applications servers.

2.4.3. 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.

2.4.4. 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.

2.4.5. 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.

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

2.4.7. 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

To enable transports to configure listening on one or more ports in a common way, this grouping is defined. This grouping defines SSH and TLS specific containers, each of which refines the default listening port appropriately. Further, each of these transport specific containers use a feature statement, enabling NETCONF servers to accurately advertise what they support.

module: ietf-netconf-server
  +--rw netconf-server
    +--rw listen
      +--rw ssh (ssh-listen)?
        |  +--rw (one-or-many)?
        |     +--:(one-port)
        |        |  +--rw port? inet:port-number
        |     +--:(many-ports)
        |        +--rw interface* [address]
        |            +--rw address inet:host
        |            +--rw port? inet:port-number
      +--rw tls (tls-listen)?
        +--rw (one-or-many)?
          +--:(one-port)
          |  +--rw port? inet:port-number
          +--:(many-ports)
            +--rw interface* [address]
            +--rw address inet:host
            +--rw port? inet:port-number

To enable transports to configure initiating connections to remote applications in a common way, this grouping is defined. This grouping configures a list of network-managers, each with some transport-specific configuration augmented in. Each of the transport specific containers use a feature statement, enabling NETCONF servers to accurately advertise what they support.

module: ietf-netconf-server
  +--rw netconf-server
3.2. YANG Module

This YANG module imports YANG types from [RFC6991].

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

<CODE BEGINS> file "ietf-netconf-server.@2014-05-16.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
  }

Watsen & Schoenwaelder  Expires December 03, 2014 [Page 7]
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.: please update the date to the date of publication

revision "2014-01-24" {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Server Configuration Model";
}

// Features

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

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 {

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

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.";
}

// Groupings

grouping one-or-many-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 NETCONF server listens on on all interfaces.";
        }
    }
    case many-ports {
        list interface {
            key "address";
            leaf address {
                type inet:host;
                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 NETCONF server listens on.";
            }
        }
    }
}

grouping network-managers-config {
    container network-managers {
        description
        "A list of network managers the device initiates connections to. The configuration for each network manager specifies its details, including its endpoints, the type of connection to maintain, and the reconnection strategy to use.";
        list network-manager {
            key name;
            leaf name {
                type string {
                    length 1..64;  // XXX why these limits?
                }
            }
        }
    }
}
mandatory true;
description
 "An arbitrary name for the network manager the device is connecting to.";
}
leaf description {
type string;
description
 "An optional description for the network manager.";
}
container endpoints {
description
 "An ordered listing of the network manager’s endpoints that the device should attempt connecting to. Defining more than one enables the device to support high-availability scenarios.";
list endpoint {
 key address;
 min-elements 1;
 ordered-by user;
 leaf address {
 type inet:host;
 mandatory true;
 description
 "The hostname or IP address 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.";
 }
}
}
container transport {
}
container connection-type {
 description
 "Indicates the network manager’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
    network manager. If the connection goes down,
    immediately start trying to reconnect to it,
    using the reconnection strategy.

    This connection type minimizes any
    manager-to-device 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 manager's
      endpoint, a message will be sent to request a
      response from the endpoint. 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
      manager's endpoint before assuming the endpoint
      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
      endpoint will be disconnected after about
      count-max * interval-secs seconds.";
    }
  }
}

case periodic-connection {
  container periodic {
    description
    "Periodically connect to network manager, 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 manager-to-device data-transfer
delay. Note that for device-to-manager data, the
data should be sent immediately, connecting to
network manager 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 network manager again. The
device MAY establish a connection before this
time if it has data it needs to send to the
network manager. 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
network manager’s endpoint 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 network manager, 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 { value 1; }
  enum last-connected { value 2; }
}
default first-listed;
description
"Specifies which of the network manager’s endpoints the device should start with when trying to connect to the network manager. 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 listen-config {
description
"Provides the configuration of the NETCONF server to open one or more ports to listen for incoming client connections.";
container ssh {
  if-feature ssh-listen;
  uses one-or-many-config {
    refine one-or-many/one-port/port {
      default 830;
    }
  }
}
refine one-or-many/many-ports/interface/port {
  default 830;
}
}
}

container tls {
  if-feature tls-listen;
  uses one-or-many-config {
    refine one-or-many/one-port/port {
      default 6513;
    }
    refine one-or-many/many-ports/interface/port {
      default 6513;
    }
  }
}
}

grouping call-home-config {
  description
  "Provides the configuration of the NETCONF call-home
  clients to connect to, the overall call-home policy,
  and the reconnect strategy.";

  uses network-managers-config {
    augment network-managers/network-manager/transport {
      container ssh {
        if-feature ssh-call-home;
        container host-keys {
          description
            "An ordered listing of the SSH host keys the
device should advertise to the network manager.";
          list host-key {
            key name;
            min-elements 1;  // requires ‘ssh’ element?
            ordered-by user;
            leaf name {
              type string;
              mandatory true;
              description
                "The name of a host key the device should
advertise during the SSH key exchange.";
            }
          }
        }
        }
      }
    }
  }
}

container tls {
4. Keep-Alives for SSH and TLS

One the objectives listed above, Keep-Alives for Persistent Connections (Section 2.4.6) indicates a need for a "keep-alive" mechanism. This section specifies how the NETCONF keep-alive mechanism is to be implemented.

Both SSH and TLS have the ability to support keep-alives. Using these mechanisms, the keep-alive messages are sent inside the encrypted tunnel, thus thwarting spoof attacks.

4.1. SSH

The SSH keep-alive solution that is expected to be used when configured using the data model defined in this document 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 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.

4.2. TLS

The TLS keep-alive solution 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 over TLS 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 conveniently when keep-alives are needed the most.

5. User Authentication for TLS

5.1. Introduction

The NETCONF Server Module defined in this draft focuses on the configuration the SSH and TLS transports. This module does not define a means to configure User Authentication, as that is a stated focus for [draft-ietf-netmod-system-mgmt], however, that draft does not define configuration nodes for TLS client authentication. Thus, this draft also includes the following YANG module to augment TLS client authentication into the "ietf-system" module defined in [draft-ietf-netmod-system-mgmt].

5.2. Data Model Overview

This data model augments the "ietf-system" module defined in [draft-ietf-netmod-system-mgmt] by adding some configuration nodes under its "/system/authentication" subtree.

module: ietf-system-tls-auth
augment /sys:system/sys:authentication:
  +--rw tls
5.3. 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].

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

<CODE BEGINS> file "ietf-system-tls-auth.@2014-05-16.yang"

module ietf-system-tls-auth {

    prefix "system-tls-auth";

    import ietf-system {  // draft-ietf-netmod-system-mgmt
        prefix "sys";
    }
    import ietf-netconf-acm {
        prefix nacm;  // RFC 6536
    }
    import ietf-yang-types {
        prefix yang;  // RFC 6991
    }
    import ietf-x509-certificate-to-name {
        prefix x509c2n;  // I-D.ietf-netconf-rfc5539bis
    }

    organization

    Watsen & Schoenwaelder Expires December 03, 2014 [Page 18]
"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>

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

description

"This module augments the ietf-system module in order to add TLS authentication configuration nodes to the 'authentication' container.

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 "2014-05-24" {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Server Configuration Model";
}
// Features

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

grouping tls-global-config {
    container trusted-ca-certs {
        description
        "A list of Certificate Authority (CA) certificates that a
        NETCONF server can use to authenticate a NETCONF client’s
        certificate. A client’s certificate is authenticated if
        its Issuer matches one of the configured trusted CA
        certificates.";
        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 one of the 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";
}

// 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 NETCONF 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 NETCONF
server uses the value from the user-name leaf in the
psk-map list as the NETCONF username. If the NETCONF
server cannot find an entry in the psk-map list, or if
the pre-shared keys do not match, then the NETCONF
server terminates the connection.";
    reference
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)";
    }

    }
}
6. 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" and "ietf-system-tls-auth" YANG modules which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write 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.

ietf-netconf-server:

- None.

ietf-system-tls-auth:

- /system/authentication/tls/psk-maps/psk-map/user-name: This leaf node contains a user name that some deployments may consider sensitive information.

- /system/authentication/tls/psk-maps/psk-map/key: This leaf node contains a shared key that remote clients use to authenticate themselves to the system. This value should not be readable or writable by anyone by default.
7. IANA Considerations

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

Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

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

name: ietf-netconf-server
prefix: ncserver
reference: RFC XXXX

name: ietf-system-tls-auth
prefix: sys-tls-auth
reference: RFC XXXX

8. 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.

9. 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.

10. References

10.1. Normative References


[draft-ietf-netmod-system-mgmt]
10.2. Informative References


Appendix A. Example: SSH Transport Configuration

```xml
<netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <listen>
    <ssh>
      <port>831</port>
    </ssh>
  </listen>
  <call-home>
    <network-managers>
      <network-manager>
        <name>config-mgr</name>
        <description>
          This entry requests the device to periodically connect to the network manager.
        </description>
        <endpoints>
          <endpoint>
            <address>config-mgr1.example.com</address>
          </endpoint>
          <endpoint>
            <address>config-mgr2.example.com</address>
          </endpoint>
        </endpoints>
        <transport>
          <ssh>
            <host-keys>
              <host-key>
                <name>ssh_host_key_cert</name>
              </host-key>
              <host-key>
                <name>ssh_host_key_cert2</name>
              </host-key>
            </host-keys>
          </ssh>
        </transport>
      </network-manager>
    </network-managers>
  </call-home>
</netconf-server>
```
Appendix B. Example: TLS Transport Configuration

```xml
<netconf-server xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-server">
  <listen>
    <tls>
      <interface>
        <address>192.0.2.1</address>
        <port>6514</port>
      </interface>
    </tls>
  </listen>
  <call-home>
    <network-managers>
      <network-manager>
        <name>log-monitor</name>
        <description>This entry requests the device to maintain a persistent connect to the network manager.</description>
        <endpoints>
          <endpoint>
            <address>log-monitor1.example.com</address>
          </endpoint>
          <endpoint>
            <address>log-monitor2.example.com</address>
          </endpoint>
        </endpoints>
        <transport><tls/></transport>
        <connection-type><persistent><keep-alives><interval-secs>5</interval-secs><count-max>3</count-max></keep-alives></persistent></connection-type>
      </network-manager>
    </network-managers>
  </call-home>
</netconf-server>
```
</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>
</network-manager>
</network-managers>
</call-home>
</netconf-server>

Appendix C. Example: TLS Authentication Configuration

<system xmlns="urn:ietf:params:xml:ns:yang:ietf-system">
  <authentication>
    <trusted-ca-certs>
      <trusted-ca-cert>
        QW4gRWFzdGVyIGVnZywgZm9yIHRob3NlIHdobyBtaWdodCBsb29rICA6KQo=
      </trusted-ca-cert>
    </trusted-ca-certs>
    <trusted-client-certs>
      <trusted-client-cert>
        SSBhbSB0aGUgZWdnIG1hbiwgdGhleSBhcmUgdGhlIGVnZzN0d2VwZW4uCg==
      </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>
</system>
<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>
</authentication>
</system>

Appendix D. Change Log

D.1. I-D to 00

  o Changed title to "NETCONF Server Configuration Model"
  o Mapped inbound/outbound to listen/call-home
  o Restructured YANG module to place transport selection deeper into
    the tree, providing a more intuitive data model
  o Added section "Keep-Alives for SSH and TLS"
  o Updated the Security Considerations section
  o Added text for supporting VRFs via augments
  o Factored the TLS-AUTH config into another module augmenting the
    "ietf-system" module

D.2. 00 to 01

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

Appendix E. Open Issues

  o 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?

Authors’ Addresses

Kent Watsen
Juniper Networks
EMail: kwatsen@juniper.net

Juergen Schoenwaelder
Jacobs University Bremen
EMail: j.schoenwaelder@jacobs-university.de