NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 9 January 2021

K. Watsen
Watsen Networks
M. Scharf
Hochschule Esslingen

8 July 2020

YANG Groupings for TCP Clients and TCP Servers
draft-ietf-netconf-tcp-client-server-07

Abstract

This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers, either as standalone or in conjunction with a stack protocol layer specific configurations.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

* "DDDD" --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

* "2020-07-08" --> the publication date of this draft

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

* Appendix A. Change Log

Status of This Memo

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

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

This Internet-Draft will expire on 9 January 2021.

Copyright Notice

Table of Contents

1. Introduction

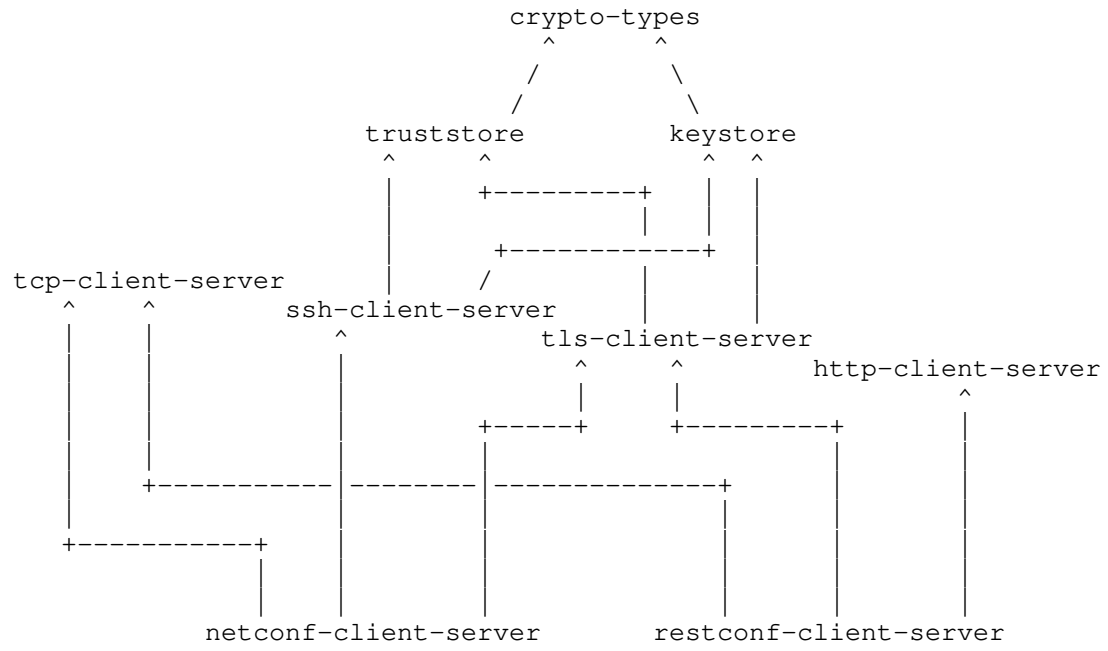
This document defines three YANG 1.1 [RFC7950] modules to support the configuration of TCP clients and TCP servers, either as standalone or in conjunction with a stack protocol layer specific configurations.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to define configuration modules for clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Links the each RFC are provided below the diagram.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

2. The "ietf-tcp-common" Module

2.1. Data Model Overview

2.1.1. Model Scope

This document defines a common "grouping" statement for basic TCP connection parameters that matter to applications. In some TCP stacks, such parameters can also directly be set by an application using system calls, such as the socket API. The base YANG model in this document focuses on modeling TCP keep-alives. This base model can be extended as needed.

2.1.2. Features

The following diagram lists all the "feature" statements defined in the "ietf-tcp-common" module:

```
Features:
  +-- keepalives-supported
```

2.1.3. Groupings

The following diagram lists all the "grouping" statements defined in the "ietf-keystore" module:

```
Groupings:
  +-- tcp-common-grouping
  +-- tcp-connection-grouping
```

Each of these groupings are presented in the following subsections.

2.1.3.1. The "tcp-common-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tcp-common-grouping" grouping:

```
  grouping tcp-common-grouping
    +-- keepalives! {keepalives-supported}?
       +-- idle-time         uint16
       +-- max-probes        uint16
       +-- probe-interval    uint16
```

Comments:

* The "keepalives" node is a "presence" node so that the decendent nodes' "mandatory true" doesn't imply that keepalives must be configured.

* The "idle-time", "max-probes", and "probe-interval" nodes have the common meanings. Please see the YANG module in Section 2.3 for details.

2.1.3.2. The "tcp-connection-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tcp-connection-grouping" grouping:

```
  grouping tcp-connection-grouping
    +---u tcp-common-grouping
```

Comments:

* This grouping uses the "tcp-common-grouping" grouping discussed in Section 2.1.3.1.

2.1.4. Protocol-accessible Nodes

The "ietf-tcp-common" module does not contain any protocol-accessible nodes.

2.1.5. Guidelines for Configuring TCP Keep-Alives

Network stacks may include "keep-alives" in their TCP implementations, although this practice is not universally accepted. If keep-alives are included, [RFC1122] [RFC793bis] mandates that the application MUST be able to turn them on or off for each TCP connection, and that they MUST default to off.

Keep-alive mechanisms exist in many protocols. Depending on the protocol stack, TCP keep-alives may only be one out of several alternatives. Which mechanism(s) to use depends on the use case and application requirements. If keep-alives are needed by an application, it is RECOMMENDED that the aliveness check happens only at the protocol layers that are meaningful to the application.

A TCP keep-alive mechanism SHOULD only be invoked in server applications that might otherwise hang indefinitely and consume resources unnecessarily if a client crashes or aborts a connection during a network failure [RFC1122]. TCP keep-alives may consume significant resources both in the network and in endpoints (e.g., battery power). In addition, frequent keep-alives risk network congestion. The higher the frequency of keep-alives, the higher the overhead.

Given the cost of keep-alives, parameters have to be configured carefully:

* The default idle interval (leaf "idle-time") MUST default to no less than two hours, i.e., 7200 seconds [RFC1122]. A lower value MAY be configured, but keep-alive messages SHOULD NOT be transmitted more frequently than once every 15 seconds. Longer intervals SHOULD be used when possible.

* The maximum number of sequential keep-alive probes that can fail (leaf "max-probes") trades off responsiveness and robustness against packet loss. ACK segments that contain no data are not reliably transmitted by TCP. Consequently, if a keep-alive mechanism is implemented it MUST NOT interpret failure to respond to any specific probe as a dead connection [RFC1122]. Typically a single-digit number should suffice.

* TCP implementations may include a parameter for the number of seconds between TCP keep-alive probes (leaf "probe-interval"). In order to avoid congestion, the time interval between probes MUST NOT be smaller than one second. Significantly longer intervals SHOULD be used. It is important to note that keep-alive probes (or replies) can get dropped due to network congestion. Sending further probe messages into a congested path after a short interval, without backing off timers, could cause harm and result in a congestion collapse. Therefore it is essential to pick a large, conservative value for this interval.

2.2. Example Usage

This section presents an example showing the "tcp-common-grouping" populated with some data.

```
   <tcp-common xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-common">
     <keepalives>
       <idle-time>15</idle-time>
       <max-probes>3</max-probes>
       <probe-interval>30</probe-interval>
     </keepalives>
   </tcp-common>
```

2.3. YANG Module

The ietf-tcp-common YANG module references [RFC6991].

```
<CODE BEGINS> file "ietf-tcp-common@2020-07-08.yang"

module ietf-tcp-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-common";
  prefix tcpcmn;

  organization
    "IETF NETCONF (Network Configuration) Working Group and the
     IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
               <http://datatracker.ietf.org/wg/tcpm/>
     WG List:  <mailto:netconf@ietf.org>
               <mailto:tcpm@ietf.org>
     Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
               Michael Scharf
               <mailto:michael.scharf@hs-esslingen.de>";

  description
    "This module defines reusable groupings for TCP commons that
     can be used as a basis for specific TCP common instances.

     Copyright (c) 2020 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
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC DDDD
     (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
```

```
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

      revision 2020-07-08 {
        description
          "Initial version";
        reference
          "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
      }

      // Features
      feature keepalives-supported {
        description
          "Indicates that keepalives are supported.";
      }

      // Groupings

      grouping tcp-common-grouping {
        description
          "A reusable grouping for configuring TCP parameters common
           to TCP connections as well as the operating system as a
           whole.";
        container keepalives {
          if-feature "keepalives-supported";
          presence
           "Indicates that keepalives are enabled.  Present so that
            the decendant nodes' 'mandatory true' doesn't imply that
            this node  must be configured.";
          description
            "Configures the keep-alive policy, to proactively test the
             aliveness of the TCP peer.  An unresponsive TCP peer is
             dropped after approximately (idle-time + max-probes
             * probe-interval) seconds.";
          leaf idle-time {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            mandatory true;
            description
              "Sets the amount of time after which if no data has been
```

```
              received from the TCP peer, a TCP-level probe message
              will be sent to test the aliveness of the TCP peer.
              Two hours (7200 seconds) is safe value, per RFC 1122.";
            reference
              "RFC 1122:
               Requirements for Internet Hosts -- Communication Layers";
          }
          leaf max-probes {
            type uint16 {
              range "1..max";
            }
            mandatory true;
            description
              "Sets the maximum number of sequential keep-alive probes
               that can fail to obtain a response from the TCP peer
               before assuming the TCP peer is no longer alive.";
          }
          leaf probe-interval {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            mandatory true;
            description
              "Sets the time interval between failed probes. The interval
               SHOULD be significantly longer than one second in order to
               avoid harm on a congested link.";
          }
        } // container keepalives
      } // grouping tcp-common-grouping


      grouping tcp-connection-grouping {
        description
          "A reusable grouping for configuring TCP parameters common
           to TCP connections.";
         uses tcp-common-grouping;
      }

    }

    <CODE ENDS>
```

3. The "ietf-tcp-client" Module

3.1. Data Model Overview

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tcp-client" module:

```
Features:
  +-- local-binding-supported
  +-- tcp-client-keepalives
  +-- proxy-connect
  +-- socks5-gss-api
  +-- socks5-username-password
```

3.1.2. Groupings

The following diagram lists all the "grouping" statements defined in the "ietf-tcp-client" module:

```
Groupings:
  +-- tcp-client-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "tcp-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tcp-client-grouping" grouping:

```
  grouping tcp-client-grouping
    +-- remote-address                    inet:host
    +-- remote-port?                      inet:port-number
    +-- local-address?                    inet:ip-address
    |       {local-binding-supported}?
    +-- local-port?                       inet:port-number
    |       {local-binding-supported}?
    +-- proxy-server! {proxy-connect}?
    |  +-- (proxy-type)
    |     +--:(socks4)
    |     |  +-- socks4-parameters
    |     |     +-- remote-address    inet:ip-address
    |     |     +-- remote-port?      inet:port-number
    |     +--:(socks4a)
    |     |  +-- socks4a-parameters
    |     |     +-- remote-address    inet:host
    |     |     +-- remote-port?      inet:port-number
    |     +--:(socks5)
    |        +-- socks5-parameters
    |           +-- remote-address               inet:host
    |           +-- remote-port?                 inet:port-number
    |           +-- authentication-parameters!
    |              +-- (auth-type)
    |                 +--:(gss-api) {socks5-gss-api}?
    |                 |  +-- gss-api
    |                 +--:(username-password)
    |                           {socks5-username-password}?
    |                    +-- username-password
    |                       +-- username?   string
    |                       +-- password?   string
    |
    +---u tcpcmn:tcp-connection-grouping
```

Comments:

* The "remote-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, a hostname.

* The "remote-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* The "local-address" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node, which is enabled by the "local-binding-supported" feature (Section 2.1.2), is not mandatory. Its default value is '0', indicating that the operating system can pick an arbitrary port number.

* The "proxy-server" node is enabled by a "feature" statement and, for servers that enable it, is a "presence" container so that the decendent "mandatory true" choice node doesn't imply that the proxt-server node must be configured.

* This grouping uses the "tcp-connection-grouping" grouping discussed in Section 2.1.3.2.

3.1.3. Protocol-accessible Nodes

The "ietf-tcp-client" module does not contain any protocol-accessible nodes.

3.2. Example Usage

This section presents an example showing the "tcp-client-grouping" populated with some data.

```
<tcp-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-client">
  <remote-address>www.example.com</remote-address>
  <remote-port>443</remote-port>
  <local-address>0.0.0.0</local-address>
  <local-port>0</local-port>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-client>
```

3.3. YANG Module

The ietf-tcp-client YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-client@2020-07-08.yang"

```
module ietf-tcp-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-client";
  prefix tcpc;

  import ietf-inet-types {
    prefix inet;
```

```
      reference
        "RFC 6991: Common YANG Data Types";
    }

    import ietf-netconf-acm {
      prefix nacm;
      reference
        "RFC 8341: Network Configuration Access Control Model";
    }

    import ietf-tcp-common {
      prefix tcpcmn;
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    organization
      "IETF NETCONF (Network Configuration) Working Group and the
       IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

    contact
      "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
                 <http://datatracker.ietf.org/wg/tcpm/>
       WG List:  <mailto:netconf@ietf.org>
                 <mailto:tcpm@ietf.org>
       Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                 Michael Scharf
                 <mailto:michael.scharf@hs-esslingen.de>";

    description
      "This module defines reusable groupings for TCP clients that
       can be used as a basis for specific TCP client instances.

       Copyright (c) 2020 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
       (https://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC DDDD
       (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
       itself for full legal notices.

       The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
```

```
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

    revision 2020-07-08 {
      description
        "Initial version";
      reference
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }

    // Features

    feature local-binding-supported {
      description
        "Indicates that the server supports configuring local
         bindings (i.e., the local address and local port) for
         TCP clients.";
    }

    feature tcp-client-keepalives {
      description
        "Per socket TCP keepalive parameters are configurable for
         TCP clients on the server implementing this feature.";
    }

    feature proxy-connect {
      description
        "Proxy connection configuration is configurable for
         TCP clients on the server implementing this feature.";
    }

    feature socks5-gss-api {
      description
        "Indicates that the server supports authenticating
         using GSSAPI when initiating TCP connections via
         and SOCKS Version 5 proxy server.";
      reference
        "RFC 1928: SOCKS Protocol Version 5";
    }

    feature socks5-username-password {
      description
        "Indicates that the server supports authenticating
         using username/password when initiating TCP
         connections via and SOCKS Version 5 proxy
```

```
          server.";
      reference
        "RFC 1928: SOCKS Protocol Version 5";
    }

    // Groupings

    grouping tcp-client-grouping {
      description
        "A reusable grouping for configuring a TCP client.

        Note that this grouping uses fairly typical descendent
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tcp-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      leaf remote-address {
        type inet:host;
        mandatory true;
        description
          "The IP address or hostname of the remote peer to
           establish a connection with.  If a domain name is
           configured, then the DNS resolution should happen on
           each connection attempt.  If the DNS resolution
           results in multiple IP addresses, the IP addresses
           are tried according to local preference order until
           a connection has been established or until all IP
           addresses have failed.";
      }
      leaf remote-port {
        type inet:port-number;
        default "0";
        description
          "The IP port number for the remote peer to establish a
           connection with.  An invalid default value (0) is used
           (instead of 'mandatory true') so that as application
           level data model may 'refine' it with an application
           specific default port number value.";
      }
      leaf local-address {
        if-feature "local-binding-supported";
        type inet:ip-address;
        description
          "The local IP address/interface (VRF?) to bind to for when
```

```
            connecting to the remote peer.  INADDR_ANY ('0.0.0.0') or
            INADDR6_ANY ('0:0:0:0:0:0:0:0' a.k.a. '::') MAY be used to
            explicitly indicate the implicit default, that the server
            can bind to any IPv4 or IPv6 addresses, respectively.";
        }
        leaf local-port {
          if-feature "local-binding-supported";
          type inet:port-number;
          default "0";
          description
            "The local IP port number to bind to for when connecting
             to the remote peer.  The port number '0', which is the
             default value, indicates that any available local port
             number may be used.";
        }

        container proxy-server {
          if-feature "proxy-connect";
          presence
            "Indicates that a proxy connection is configured.
             Present so that the 'proxy-type' node's 'mandatory
             true' doesn't imply that the proxy connection
             must be configured.";
          choice proxy-type {
            mandatory true;
            description
              "Selects a proxy connection protocol.";
            case socks4 {
              container socks4-parameters {
                leaf remote-address {
                  type inet:ip-address;
                  mandatory true;
                  description
                    "The IP address of the proxy server.";
                }
                leaf remote-port {
                  type inet:port-number;
                  default "1080";
                  description
                    "The IP port number for the proxy server.";
                }
                description
                  "Parameters for connecting to a TCP-based proxy
                   server using the SOCKS4 protocol.";
                reference
                  "SOCKS, Proceedings: 1992 Usenix Security Symposium.";
              }
            }
```

```
            case socks4a {
              container socks4a-parameters {
                leaf remote-address {
                  type inet:host;
                  mandatory true;
                  description
                    "The IP address or hostname of the proxy server.";
                }
                leaf remote-port {
                  type inet:port-number;
                  default "1080";
                  description
                    "The IP port number for the proxy server.";
                }
                description
                  "Parameters for connecting to a TCP-based proxy
                   server using the SOCKS4a protocol.";
                reference
                  "SOCKS Proceedings:
                     1992 Usenix Security Symposium.
                   OpenSSH message:
                     SOCKS 4A: A Simple Extension to SOCKS 4 Protocol
                     https://www.openssh.com/txt/socks4a.protocol";
              }
            }
            case socks5 {
              container socks5-parameters {
                leaf remote-address {
                  type inet:host;
                  mandatory true;
                  description
                    "The IP address or hostname of the proxy server.";
                }
                leaf remote-port {
                  type inet:port-number;
                  default "1080";
                  description
                    "The IP port number for the proxy server.";
                }
                container authentication-parameters {
                  presence
                    "Indicates that an authentication mechanism
                     has been configured.  Present so that the
                     'auth-type' node's 'mandatory true' doesn't
                     imply that an authentication mechanism
                     must be configured.";
                  description
                    "A container for SOCKS Version 5 authentication
```

```
          mechanisms.

          A complete list of methods is defined at:
          https://www.iana.org/assignments/socks-methods
          /socks-methods.xhtml.";
        reference
          "RFC 1928: SOCKS Protocol Version 5";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst supported SOCKS Version 5
             authentication mechanisms.";
          case gss-api {
            if-feature socks5-gss-api;
            container gss-api {
              description
                "Contains GSS-API configuration.  Defines
                 as an empty container to enable specific
                 GSS-API configuration to be augmented in
                 by future modules.";
              reference
                "RFC 1928: SOCKS Protocol Version 5
                 RFC 2743: Generic Security Service
                           Application Program Interface
                           Version 2, Update 1";
            }
          }
          case username-password {
            if-feature socks5-username-password;
            container username-password {
              leaf username {
                type string;
                description
                  "The 'username' value to use.";
              }
              leaf password {
                nacm:default-deny-all;
                type string;
                description
                  "The 'password' value to use.";
              }
              description
                "Contains Username/Password configuration.";
              reference
                "RFC 1929: Username/Password Authentication
                           for SOCKS V5";
            }
          }
```

```
                }
              }
              description
                "Parameters for connecting to a TCP-based proxy server
                 using the SOCKS5 protocol.";
              reference
                "RFC 1928: SOCKS Protocol Version 5";
            }
          }
        }
        description
          "Proxy server settings.";
      }

      uses tcpcmn:tcp-connection-grouping {
        augment "keepalives" {
          if-feature "tcp-client-keepalives";
          description
            "Add an if-feature statement so that implementations
             can choose to support TCP client keepalives.";
        }
      }
    }
  }
```

<CODE ENDS>

4. The "ietf-tcp-server" Module

4.1. Data Model Overview

4.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tcp-server" module:

```
Features:
  +-- tcp-server-keepalives
```

4.1.2. Groupings

The following diagram lists all the "grouping" statements defined in the "ietf-tcp-server" module:

```
Groupings:
  +-- tcp-server-grouping
```

Each of these groupings are presented in the following subsections.

4.1.2.1. The "tcp-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tcp-server-grouping" grouping:

```
  grouping tcp-server-grouping
    +-- local-address                    inet:ip-address
    +-- local-port?                      inet:port-number
    +---u tcpcmn:tcp-connection-grouping
```

Comments:

* The "local-address" node, which is mandatory, may be configured as an IPv4 address, an IPv6 address, or a wildcard value.

* The "local-port" node is not mandatory, but its default value is the invalid value '0', thus forcing the consuming data model to refine it in order to provide it an appropriate default value.

* This grouping uses the "tcp-connection-grouping" grouping discussed in Section 2.1.3.2.

4.1.3. Protocol-accessible Nodes

The "ietf-tcp-server" module does not contain any protocol-accessible nodes.

4.2. Example Usage

This section presents an example showing the "tcp-server-grouping" populated with some data.

```
<tcp-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-server">
  <local-address>10.20.30.40</local-address>
  <local-port>7777</local-port>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-server>
```

4.3. YANG Module

The ietf-tcp-server YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-server@2020-07-08.yang"

```
   module ietf-tcp-server {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-server";
     prefix tcps;

     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     import ietf-tcp-common {
       prefix tcpcmn;
       reference
         "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group and the
        IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

     contact
       "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
                  <http://datatracker.ietf.org/wg/tcpm/>
        WG List:  <mailto:netconf@ietf.org>
                  <mailto:tcpm@ietf.org>
        Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
                  Michael Scharf
                  <mailto:michael.scharf@hs-esslingen.de>";

     description
       "This module defines reusable groupings for TCP servers that
        can be used as a basis for specific TCP server instances.

        Copyright (c) 2020 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
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC DDDD
        (https://www.rfc-editor.org/info/rfcDDDD); see the RFC
        itself for full legal notices.
```

```
        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

      revision 2020-07-08 {
        description
          "Initial version";
        reference
          "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
      }

      // Features

      feature tcp-server-keepalives {
        description
          "Per socket TCP keepalive parameters are configurable for
           TCP servers on the server implementing this feature.";
      }


      // Groupings

      grouping tcp-server-grouping {
        description
          "A reusable grouping for configuring a TCP server.

           Note that this grouping uses fairly typical descendent
           node names such that a stack of 'uses' statements will
           have name conflicts.  It is intended that the consuming
           data model will resolve the issue (e.g., by wrapping
           the 'uses' statement in a container called
           'tcp-server-parameters').  This model purposely does
           not do this itself so as to provide maximum flexibility
           to consuming models.";
        leaf local-address {
          type inet:ip-address;
          mandatory true;
          description
            "The local IP address to listen on for incoming
             TCP client connections.  INADDR_ANY (0.0.0.0) or
             INADDR6_ANY (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be
             used when the server is to listen on all IPv4 or
             IPv6 addresses, respectively.";
        }
        leaf local-port {
```

```
        type inet:port-number;
        default "0";
        description
          "The local port number to listen on for incoming TCP
           client connections.  An invalid default value (0)
           is used (instead of 'mandatory true') so that an
           application level data model may 'refine' it with
           an application specific default port number value.";
      }
      uses tcpcmn:tcp-connection-grouping {
        augment "keepalives" {
          if-feature "tcp-server-keepalives";
          description
            "Add an if-feature statement so that implementations
             can choose to support TCP server keepalives.";
        }
      }
    }
  }
```

<CODE ENDS>

5. Security Considerations

5.1. The "ietf-tcp-common" YANG Module

The "ietf-tcp-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. The "ietf-tcp-client" YANG Module

The "ietf-tcp-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

One readable data node defined in this YANG module may be considered sensitive or vulnerable in some network environments. This node is as follows:

* The "proxy-server/socks5-parameters/authentication-parameters/username-password/password" node:

  The cleartext "password" node defined in the "tcp-client-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.3. The "ietf-tcp-server" YANG Module

The "ietf-tcp-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

6.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         ietf-tcp-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-common
prefix:       tcpcmn
reference:    RFC DDDD

name:         ietf-tcp-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-client
prefix:       tcpc
reference:    RFC DDDD

name:         ietf-tcp-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tcp-server
prefix:       tcps
reference:    RFC DDDD
```

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

7.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-15, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-15>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-03, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-03>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-17, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-17>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-19>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-19>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-19>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-06>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-19, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-19>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-10>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Change Log

This section is to be removed before publishing as an RFC.

A.1. 00 to 01

* Added 'local-binding-supported' feature to TCP-client model.

* Added 'keepalives-supported' feature to TCP-common model.

* Added 'external-endpoint-values' container and 'external-endpoints' feature to TCP-server model.

A.2. 01 to 02

* Removed the 'external-endpoint-values' container and 'external-endpoints' feature from the TCP-server model.

A.3. 02 to 03

* Moved the common model section to be before the client and server specific sections.

* Added sections "Model Scope" and "Usage Guidelines for Configuring TCP Keep-Alives" to the common model section.

A.4. 03 to 04

* Fixed a few typos.

A.5. 04 to 05

* Removed commented out "grouping tcp-system-grouping" statement kept for reviewers.

* Added a "Note to Reviewers" note to first page.

A.6. 05 to 06

* Added support for TCP proxies.

A.7. 06 to 07

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Michael Scharf
Hochschule Esslingen - University of Applied Sciences

Email: michael.scharf@hs-esslingen.de