TCPM
Internet-Draft
Intended status: Standards Track
Expires: April 14, 2019

M. Scharf
Hochschule Esslingen

October 11, 2018

Transmission Control Protocol (TCP) YANG Model
draft-scharf-tcpm-yang-tcp-00

Abstract

This document specifies a base YANG model for TCP on devices that are configured by network management protocols. The YANG model follows the standard TCP-MIB [RFC4022].

Status of This Memo

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

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

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

This Internet-Draft will expire on April 14, 2019.

Table of Contents

1. Introduction

The Transmission Control Protocol (TCP) [RFC0793] is used by several control and management protocols in the Internet. Therefore, TCP is implemented on network elements that can be configured via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. This document specifies a YANG data model [RFC6020][RFC7950] for configuring and operating TCP on network elements that support YANG data models.

Many protocol stacks on Internet hosts use other methods to configure TCP, such as operating system configuration or policies. Such TCP stacks typically cannot be configured by network management protocols such as NETCONF or RESTCONF. This document only applies to network elements that use YANG data models.

This document is related to other standardization efforts. A Management Information Base (MIB) for the Transmission Control Protocol (TCP) has been standardized [RFC4022], and various managed network devices support the TCP-MIB. A MIB providing extended statistics for TCP is also available [RFC4898]. In addition, there are also MIBs for UDP [RFC4113] and SCTP [RFC3873].

This documents defines a YANG data model for configuration and operational state of a TCP implementation. The model focuses on fundamental and standard TCP functions. The model can be augmented to address more advanced or implementation-specific TCP features.

1.1. Why a YANG Model for TCP?

[[Editor's node: This section shall be removed or reworded in later versions of the document.]]

This section summarizes reasons why the specification of a YANG model for TCP could be useful to maintain TCP's utility.

As more and more network elements can be managed by management protocols such as NETCONF or RESTCONF, network elements may need a YANG model for TCP stack configuration and operation. This could in particular apply to network elements that support the TCP-MIB [RFC4022] but migrate to management by NETCONF or RESTCONF. In such cases, one option would be to translate the TCP-MIB into a YANG model, for instance using the translation described in [RFC6643]. However, such a translated YANG model neither allows configuration, nor is the content up-to-date. For instance, the TCP-MIB refers to standards that have been obsoleted.

Given the advantages of YANG, there are also many ongoing efforts to define YANG models. An increasing number of YANG models include TCP and TCP-related parameters for specific usage scenarios. Examples are:

- TCP connection properties such as use of keep-alives can be configured by [I-D.ietf-netconf-netconf-client-server].
- TCP header attributes are modeled in [I-D.ietf-netmod-acl-model].
- TCP-related configuration of a NAT is defined in [I-D.ietf-opsawg-nat-yang].

It is possible that further YANG models would include aspects of or relate to TCP stack configuration. An example could be YANG models for other TCP-based protocols running on network elements. Instead of specifying TCP configuration separately for each use cases, a better alternative may be to define a common YANG model for TCP, from which definitions and objects could be reused as needed.

The intention of this document is to explore whether a YANG model for TCP is useful, and if so, what such a model would include (and what not).

This document targets the TCPM working group as the TCPM charter includes work on "interfaces that maintain TCP's utility". The YANG model deals with the network management interface of a TCP implementation. The interface used for data transfer between an application and the TCP stack is outside the scope of this document.

1.2. Open Issues

[[Editor's node: This section shall be removed in later versions of the document.]]

There are many open questions on the scope of this document that need discussion and community feedback:

- Scope: TCP stacks can typically be customized by configuration, including implementation-specific internal behavior. A standard YANG model should focus on fundamental TCP parameters that are independent of the internals of an implementation, that are available in all or most implementations, and that matter for network management. This set of TCP configuration parameters needs to be determined. Additional implementation-specific configuration could be added by augmentation of the YANG model and would not require standardization.

- Backward compatibility: There may be implementations that support the TCP-MIB [RFC4022], possibly in addition to YANG models. In such cases, one option could be to translate the TCP-MIB [RFC4022] into an equivalent YANG model. The TCP-MIB is not up-to-date. Additions could be needed to reflect e.g. the progress of TCP standards. It is an open question whether a YANG model would need "backward-compatibility" to TCP-MIB entries, and, if so, if this is needed for all TCP-MIB entries.

- General definitions: A TCP model could perhaps include general YANG type definitions and groupings for TCP. They could be reused by other YANG models. Having a common definition of TCP-related attributes could ensure consistency. It is an open question whether definitions in a TCP model would indeed be used by other YANG models.

- Extended statistics: It needs to be decided whether extended statistics [RFC4898] would be in scope of a TCP YANG model.

- Suggested MIB additions: Extensions to the TCP-MIB [RFC4022] have been suggested, e.g., in [RFC5482]. A YANG model could possibly take into account such proposed extensions.

- Cross-layer functionality: The TCP configuration may have dependencies on other protocols. It needs to be figured out if and how this can be modelled. An example could be enabling of TCP keep-alives, which should be off by default [RFC1122].

- Alignment: Further discussion is needed on alignment of a TCP YANG model with other transport protocols. This would include UDP and SCTP.

- Status of [RFC4022]: Standardization of a YANG model does not affect an existing MIB. As a result, updating or deprecating [RFC4022] may not be necessary even if a YANG model for TCP is standardized. However, more analysis is needed on how future versions of this I-D would relate to or reference [RFC4022].

- Implementation aspects: Implementation aspects of a YANG model for TCP need further analysis.

- Authoring: If there is a need for similarity to the TCP-MIB, authors and contributors to [RFC4022] could/should be added as authors or contributors to this document.

2. Requirements 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.

3. Overview of the YANG Model for TCP

[[Editor's node: This section should to be improved in follow-up versions of this document.]]

3.1. Model Design

[[Editor's node: In potential follow-up versions of this document, this section will discuss the design of the TCP YANG model.]]

A standard YANG data model for TCP should follow the Network Management Datastore Architecture (NMDA) [RFC8342].

The YANG model in the -00 version of this document has been translated from the TCP-MIB [RFC4022] without modifications (apart from removing outdated meta-data). The motivation of using the standard TCP-MIB definitions as a baseline for -00 is to facilitate the tracking of potential changes in later versions of this specification, as far as this is needed.

The TCP-MIB defined in [RFC4022] consists of several objects and two tables:

- Parameters of a TCP protocol engine. These include parameters such as the retransmission algorithm in use and the retransmission timeout values.

- Statistics of a TCP protocol engine. These include counters for the number of active/passive opens, input/output segments, and errors.

- A TCP connection table provides access to status information for all TCP connections handled by a TCP protocol engine. In addition, the table reports identification of the operating system level processes that handle the TCP connections.

- A TCP listener table provides access to information about all TCP listening endpoints known by a TCP protocol engine. And as with the connection table, this table also reports the identification of the operating system level processes that handle this listening TCP endpoint.

The extended statistics defined in [RFC4898] have been omitted in the initial version of this specification. They can be added as needed.

3.2. Tree Diagram

[[Editor's node: This section is TBD.]]

```
  module: TCP-MIB
    +--rw tcp
       +--ro tcpRtoAlgorithm?      enumeration
       +--ro tcpRtoMin?            int32
       +--ro tcpRtoMax?            int32
       +--ro tcpMaxConn?           int32
       +--ro tcpActiveOpens?       yang:counter32
       +--ro tcpPassiveOpens?      yang:counter32
       +--ro tcpAttemptFails?      yang:counter32
       +--ro tcpEstabResets?       yang:counter32
       +--ro tcpCurrEstab?         yang:gauge32
       +--ro tcpInSegs?            yang:counter32
       +--ro tcpOutSegs?           yang:counter32
       +--ro tcpRetransSegs?       yang:counter32
       x--rw tcpConnEntry* [tcpConnLocalAddress tcpConnLocalPort
tcpConnRemAddress tcpConnRemPort]
       |  x--rw tcpConnState?          enumeration
       |  x--ro tcpConnLocalAddress    inet:ipv4-address
       |  x--ro tcpConnLocalPort       int32
       |  x--ro tcpConnRemAddress      inet:ipv4-address
       |  x--ro tcpConnRemPort         int32
       +--ro tcpInErrs?            yang:counter32
       +--ro tcpOutRsts?           yang:counter32
       +--ro tcpHCInSegs?          yang:counter64
       +--ro tcpHCOutSegs?         yang:counter64
       +--rw tcpConnectionEntry* [tcpConnectionLocalAddressType
tcpConnectionLocalAddress tcpConnectionLocalPort tcpConnectionRemAddressType
tcpConnectionRemAddress tcpConnectionRemPort]
       |  +--ro tcpConnectionLocalAddressType    inet-address:InetAddressType
       |  +--ro tcpConnectionLocalAddress        inet-address:InetAddress
       |  +--ro tcpConnectionLocalPort           inet-address:InetPortNumber
       |  +--ro tcpConnectionRemAddressType      inet-address:InetAddressType
       |  +--ro tcpConnectionRemAddress          inet-address:InetAddress
       |  +--ro tcpConnectionRemPort             inet-address:InetPortNumber
       |  +--rw tcpConnectionState?              enumeration
       |  +--ro tcpConnectionProcess?            uint32
       +--rw tcpListenerEntry* [tcpListenerLocalAddressType
tcpListenerLocalAddress tcpListenerLocalPort]
          +--ro tcpListenerLocalAddressType    inet-address:InetAddressType
          +--ro tcpListenerLocalAddress        inet-address:InetAddress
          +--ro tcpListenerLocalPort           inet-address:InetPortNumber
          +--ro tcpListenerProcess?            uint32
```

4. TCP YANG Model

[[Editor's node: This section is TBD.]]

The following YANG module has been generated by smidump 0.4.8 using

the command "smidump -f yang TCP-MIB". Improvements to the translated YANG model are planed for future versions of this specification. The -00 version of this model is just published as a

baseline and to enable tracking of changes in later versions of the memo.

```
<CODE BEGINS> file "ietf-tcp-translated@2005-02-18.yang"
module TCP-MIB {

  /*** NAMESPACE / PREFIX DEFINITION ***/

  namespace "urn:ietf:params:xml:ns:yang:smiv2:TCP-MIB";
  prefix "tcp-mib";

  /*** LINKAGE (IMPORTS / INCLUDES) ***/

  import INET-ADDRESS-MIB { prefix "inet-address"; }
  import inet-types       { prefix "inet"; }
  import yang-types       { prefix "yang"; }

  /*** META INFORMATION ***/

  organization
   "TBD";

  contact
   "TBD";

  description
   "TBD";

  revision "2005-02-18" {
    description
     "IP version neutral revision, published as RFC 4022.";
  }
  revision "1994-11-01" {
    description
     "Initial SMIv2 version, published as RFC 2012.";
  }
  revision "1991-03-31" {
    description
     "The initial revision of this MIB module was part of
      MIB-II.";
  }

  container tcp {

    leaf tcpRtoAlgorithm {
      type enumeration {
        enum other    { value 1; }
        enum constant { value 2; }
```

```
          enum rsre     { value 3; }
          enum vanj     { value 4; }
          enum rfc2988  { value 5; }
        }
        config false;
        description
         "The algorithm used to determine the timeout value used for
          retransmitting unacknowledged octets.";
      }

      leaf tcpRtoMin {
        type int32 {
          range "0..2147483647";
        }
        units "milliseconds";
        config false;
        description
         "The minimum value permitted by a TCP implementation for
          the retransmission timeout, measured in milliseconds.
          More refined semantics for objects of this type depend
          on the algorithm used to determine the retransmission
          timeout; in particular, the IETF standard algorithm
          rfc2988(5) provides a minimum value.";
      }

      leaf tcpRtoMax {
        type int32 {
          range "0..2147483647";
        }
        units "milliseconds";
        config false;
        description
         "The maximum value permitted by a TCP implementation for
          the retransmission timeout, measured in milliseconds.
          More refined semantics for objects of this type depend
          on the algorithm used to determine the retransmission
          timeout; in particular, the IETF standard algorithm
          rfc2988(5) provides an upper bound (as part of an
          adaptive backoff algorithm).";
      }

      leaf tcpMaxConn {
        type int32 {
          range "-1..2147483647";
        }
        config false;
        description
         "The limit on the total number of TCP connections the entity
```

```
          can support.  In entities where the maximum number of
          connections is dynamic, this object should contain the
          value -1.";
      }

      leaf tcpActiveOpens {
        type yang:counter32;
        config false;
        description
         "The number of times that TCP connections have made a direct
          transition to the SYN-SENT state from the CLOSED state.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
      }

      leaf tcpPassiveOpens {
        type yang:counter32;
        config false;
        description
         "The number of times TCP connections have made a direct
          transition to the SYN-RCVD state from the LISTEN state.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
      }

      leaf tcpAttemptFails {
        type yang:counter32;
        config false;
        description
         "The number of times that TCP connections have made a direct
          transition to the CLOSED state from either the SYN-SENT
          state or the SYN-RCVD state, plus the number of times that
          TCP connections have made a direct transition to the
          LISTEN state from the SYN-RCVD state.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
      }

      leaf tcpEstabResets {
        type yang:counter32;
        config false;
        description
         "The number of times that TCP connections have made a direct
          transition to the CLOSED state from either the ESTABLISHED
          state or the CLOSE-WAIT state.
```

```
          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
      }

      leaf tcpCurrEstab {
        type yang:gauge32;
        config false;
        description
         "The number of TCP connections for which the current state
          is either ESTABLISHED or CLOSE-WAIT.";
      }

      leaf tcpInSegs {
        type yang:counter32;
        config false;
        description
         "The total number of segments received, including those
          received in error.  This count includes segments received
          on currently established connections.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
      }

      leaf tcpOutSegs {
        type yang:counter32;
        config false;
        description
         "The total number of segments sent, including those on
          current connections but excluding those containing only
          retransmitted octets.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
      }

      leaf tcpRetransSegs {
        type yang:counter32;
        config false;
        description
         "The total number of segments retransmitted; that is, the
          number of TCP segments transmitted containing one or more
          previously transmitted octets.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
      }
```

```
      /* XXX table comments here XXX */

      list tcpConnEntry {

        key "tcpConnLocalAddress tcpConnLocalPort
             tcpConnRemAddress tcpConnRemPort";
        status deprecated;
        description
         "A conceptual row of the tcpConnTable containing information
          about a particular current IPv4 TCP connection.  Each row
          of this table is transient in that it ceases to exist when
          (or soon after) the connection makes the transition to the
          CLOSED state.";


        leaf tcpConnState {
          type enumeration {
            enum closed      { value 1; }
            enum listen      { value 2; }
            enum synSent     { value 3; }
            enum synReceived { value 4; }
            enum established { value 5; }
            enum finWait1    { value 6; }
            enum finWait2    { value 7; }
            enum closeWait   { value 8; }
            enum lastAck     { value 9; }
            enum closing     { value 10; }
            enum timeWait    { value 11; }
            enum deleteTCB   { value 12; }
          }
          config true;
          status deprecated;
          description
           "The state of this TCP connection.

            The only value that may be set by a management station is
            deleteTCB(12).  Accordingly, it is appropriate for an agent
            to return a `badValue' response if a management station
            attempts to set this object to any other value.

            If a management station sets this object to the value
            deleteTCB(12), then the TCB (as defined in [RFC793]) of
            the corresponding connection on the managed node is
            deleted, resulting in immediate termination of the
            connection.

            As an implementation-specific option, a RST segment may be
            sent from the managed node to the other TCP endpoint (note,
```

```
            however, that RST segments are not sent reliably).";
        }

        leaf tcpConnLocalAddress {
          type inet:ipv4-address;
          config false;
          status deprecated;
          description
           "The local IP address for this TCP connection.  In the case
            of a connection in the listen state willing to
            accept connections for any IP interface associated with the
            node, the value 0.0.0.0 is used.";
        }

        leaf tcpConnLocalPort {
          type int32 {
            range "0..65535";
          }
          config false;
          status deprecated;
          description
           "The local port number for this TCP connection.";
        }

        leaf tcpConnRemAddress {
          type inet:ipv4-address;
          config false;
          status deprecated;
          description
           "The remote IP address for this TCP connection.";
        }

        leaf tcpConnRemPort {
          type int32 {
            range "0..65535";
          }
          config false;
          status deprecated;
          description
           "The remote port number for this TCP connection.";
        }
      }

      leaf tcpInErrs {
        type yang:counter32;
        config false;
        description
         "The total number of segments received in error (e.g., bad
```

```
          TCP checksums).

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
        }

        leaf tcpOutRsts {
          type yang:counter32;
          config false;
          description
           "The number of TCP segments sent containing the RST flag.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
        }

        leaf tcpHCInSegs {
          type yang:counter64;
          config false;
          description
           "The total number of segments received, including those
            received in error.  This count includes segments received


          on currently established connections.  This object is
          the 64-bit equivalent of tcpInSegs.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
        }

        leaf tcpHCOutSegs {
          type yang:counter64;
          config false;
          description
           "The total number of segments sent, including those on
            current connections but excluding those containing only
            retransmitted octets.  This object is the 64-bit
            equivalent of tcpOutSegs.

          Discontinuities in the value of this counter are
          indicated via discontinuities in the value of sysUpTime.";
        }


        /* XXX table comments here XXX */

        list tcpConnectionEntry {
```

```
        key "tcpConnectionLocalAddressType
             tcpConnectionLocalAddress tcpConnectionLocalPort
             tcpConnectionRemAddressType tcpConnectionRemAddress
             tcpConnectionRemPort";
        description
         "A conceptual row of the tcpConnectionTable containing
          information about a particular current TCP connection.
          Each row of this table is transient in that it ceases to
          exist when (or soon after) the connection makes the
          transition to the CLOSED state.";


        leaf tcpConnectionLocalAddressType {
          type inet-address:InetAddressType;
          config false;
          description
           "The address type of tcpConnectionLocalAddress.";
        }

        leaf tcpConnectionLocalAddress {
          type inet-address:InetAddress;
          config false;
          description
           "The local IP address for this TCP connection.  The type
            of this address is determined by the value of
            tcpConnectionLocalAddressType.

            As this object is used in the index for the
            tcpConnectionTable, implementors should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; otherwise the information
            cannot be accessed by using SNMPv1, SNMPv2c, or SNMPv3.";
        }

        leaf tcpConnectionLocalPort {
          type inet-address:InetPortNumber;
          config false;
          description
           "The local port number for this TCP connection.";
        }

        leaf tcpConnectionRemAddressType {
          type inet-address:InetAddressType;
          config false;
          description
           "The address type of tcpConnectionRemAddress.";
        }
```

```
        leaf tcpConnectionRemAddress {
          type inet-address:InetAddress;
          config false;
          description
           "The remote IP address for this TCP connection.  The type
            of this address is determined by the value of
            tcpConnectionRemAddressType.

            As this object is used in the index for the
            tcpConnectionTable, implementors should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; otherwise the information
            cannot be accessed by using SNMPv1, SNMPv2c, or SNMPv3.";
        }

        leaf tcpConnectionRemPort {
          type inet-address:InetPortNumber;
          config false;
          description
           "The remote port number for this TCP connection.";
        }

        leaf tcpConnectionState {
          type enumeration {
            enum closed      { value 1; }
            enum listen      { value 2; }
            enum synSent     { value 3; }
            enum synReceived { value 4; }
            enum established { value 5; }
            enum finWait1    { value 6; }
            enum finWait2    { value 7; }
            enum closeWait   { value 8; }
            enum lastAck     { value 9; }
            enum closing     { value 10; }
            enum timeWait    { value 11; }
            enum deleteTCB   { value 12; }
          }
          config true;
          description
           "The state of this TCP connection.

            The value listen(2) is included only for parallelism to the
            old tcpConnTable and should not be used.  A connection in
            LISTEN state should be present in the tcpListenerTable.

            The only value that may be set by a management station is
            deleteTCB(12).  Accordingly, it is appropriate for an agent
            to return a `badValue' response if a management station
```

```
            attempts to set this object to any other value.

            If a management station sets this object to the value
            deleteTCB(12), then the TCB (as defined in [RFC793]) of
            the corresponding connection on the managed node is
            deleted, resulting in immediate termination of the
            connection.

            As an implementation-specific option, a RST segment may be
            sent from the managed node to the other TCP endpoint (note,
            however, that RST segments are not sent reliably).";
        }

        leaf tcpConnectionProcess {
          type uint32;
          config false;
          description
           "The system's process ID for the process associated with
            this connection, or zero if there is no such process.  This
            value is expected to be the same as HOST-RESOURCES-MIB::
            hrSWRunIndex or SYSAPPL-MIB::sysApplElmtRunIndex for some
            row in the appropriate tables.";
        }
      }


      /* XXX table comments here XXX */

      list tcpListenerEntry {

        key "tcpListenerLocalAddressType tcpListenerLocalAddress
             tcpListenerLocalPort";
        description
         "A conceptual row of the tcpListenerTable containing
          information about a particular TCP listener.";


        leaf tcpListenerLocalAddressType {
          type inet-address:InetAddressType;
          config false;
          description
           "The address type of tcpListenerLocalAddress.  The value
            should be unknown (0) if connection initiations to all
            local IP addresses are accepted.";
        }

        leaf tcpListenerLocalAddress {
          type inet-address:InetAddress;
```

```
          config false;
          description
           "The local IP address for this TCP connection.

            The value of this object can be represented in three
            possible ways, depending on the characteristics of the
            listening application:

            1. For an application willing to accept both IPv4 and
               IPv6 datagrams, the value of this object must be
               ''h (a zero-length octet-string), with the value
               of the corresponding tcpListenerLocalAddressType
               object being unknown (0).

            2. For an application willing to accept only IPv4 or
               IPv6 datagrams, the value of this object must be
               '0.0.0.0' or '::' respectively, with
               tcpListenerLocalAddressType representing the
               appropriate address type.

            3. For an application which is listening for data
               destined only to a specific IP address, the value
               of this object is the specific local address, with
               tcpListenerLocalAddressType representing the
               appropriate address type.

            As this object is used in the index for the
            tcpListenerTable, implementors should be
            careful not to create entries that would result in OIDs
            with more than 128 subidentifiers; otherwise the information
            cannot be accessed, using SNMPv1, SNMPv2c, or SNMPv3.";
        }

        leaf tcpListenerLocalPort {
          type inet-address:InetPortNumber;
          config false;
          description
           "The local port number for this TCP connection.";
        }

        leaf tcpListenerProcess {
          type uint32;
          config false;
          description
           "The system's process ID for the process associated with
            this listener, or zero if there is no such process.  This
            value is expected to be the same as HOST-RESOURCES-MIB::
            hrSWRunIndex or SYSAPPL-MIB::sysApplElmtRunIndex for some
```

```
            row in the appropriate tables.";
        }
      }
    }

  } /* end of module TCP-MIB */
  <CODE ENDS>
```

5. IANA Considerations

[[Editor's node: This section will be completed in follow-up versions of this document.]]

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

7. References

7.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[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>.

[RFC4022] Raghunarayan, R., Ed., "Management Information Base for the Transmission Control Protocol (TCP)", RFC 4022, DOI 10.17487/RFC4022, March 2005, <https://www.rfc-editor.org/info/rfc4022>.

[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>.

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

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6643] Schoenwaelder, J., "Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules", RFC 6643, DOI 10.17487/RFC6643, July 2012, <https://www.rfc-editor.org/info/rfc6643>.

[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>.

[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>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", draft-ietf-netconf-netconf-client-server-07 (work in progress), September 2018.

[I-D.ietf-netmod-acl-model]
Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "Network Access Control List (ACL) YANG Data Model", draft-ietf-netmod-acl-model-20 (work in progress), October 2018.

[I-D.ietf-opsawg-nat-yang]
Boucadair, M., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", draft-ietf-opsawg-nat-yang-17 (work in progress), September 2018.

[LIBSMI] University of Braunschweig, "libsmi - A Library to Access SMI MIB Information", 2014, <https://www.ibr.cs.tu-bs.de/projects/libsmi/>.

[RFC3873] Pastor, J. and M. Belinchon, "Stream Control Transmission Protocol (SCTP) Management Information Base (MIB)", RFC 3873, DOI 10.17487/RFC3873, September 2004, <https://www.rfc-editor.org/info/rfc3873>.

[RFC4113] Fenner, B. and J. Flick, "Management Information Base for the User Datagram Protocol (UDP)", RFC 4113, DOI 10.17487/RFC4113, June 2005, <https://www.rfc-editor.org/info/rfc4113>.

[RFC4898] Mathis, M., Heffner, J., and R. Raghunarayan, "TCP Extended Statistics MIB", RFC 4898, DOI 10.17487/RFC4898, May 2007, <https://www.rfc-editor.org/info/rfc4898>.

[RFC5482] Eggert, L. and F. Gont, "TCP User Timeout Option", RFC 5482, DOI 10.17487/RFC5482, March 2009, <https://www.rfc-editor.org/info/rfc5482>.

Appendix A. Acknowledgements

The YANG model used in version -00 of the draft has been converted from [RFC4022] by the LIBSMI library [LIBSMI].

Author's Address

Michael Scharf
Hochschule Esslingen - University of Applied Sciences
Flandernstr. 11
Esslingen 73732
Germany

Email: michael.scharf@hs-esslingen.de