YANG Groupings for TCP Clients and TCP Servers

draft-ietf-netconf-tcp-client-server-02

• Kent Watsen

• Michael Scharf

NETCONF WG
IETF 105 (Montreal)
It is understood that TCP can work with simultaneous opens
  • such that there is effectively no difference between client and server

That said, the terms "client" and "server" herein primarily mean:
  • **Client**: the initiator of the protocol exchange
  • **Server**: the receiver of the protocol exchange

However, only when discussing NETCONF and RESTCONF:
  • **Client**: the controller/NMS application
  • **Server**: the NE/device being managed
History

NETCONF WG has been working on this problem for ~5 years

The "problem" being:

How to configure
NETCONF and RESTCONF
clients and servers?

(Using YANG Data Models)
What are YANG Data Models?

YANG is an IETF replacement for SNMP MIBs.

YANG is a way to describe data.

YANG is to NETCONF/RESTCONF and XML/JSON:

• as MIB is to SNMP
• as XSD is to XML
• as ASN.1 is PEM/DER/BER
• as ABNF is to binary data.
NETCONF WG had the following hierarchy of adopted WG drafts.

- The NETCONF protocol has transport bindings to both SSH and TLS.
- The RESTCONF protocol has a transport binding to HTTP/TLS.
The YANG models defined knobs for configuring SSH and TLS level keepalives.

Nokia and the Broadband Forum (BBF) implementation
- A replacement for TR-069 (i.e., how cable modems call home).
- Discovered that TLS-level keepalives were not supported in OpenSSL.

Hence a request to add knobs to configure TCP-level keepalives.

But how?
NETCONF WG discussed creating explicit TCP, HTTP, and HTTPS layers.

- The idea being that then each layer could define its own keepalive config parameters...
Circa IETF 104 (Current)

Moving from an "is-a" to a "has-a" relationship
- enables arbitrary protocol stack compositions (e.g., tcp/http vs. tcp/tls/http)
Draft defines 3 YANG modules

**module: ietf-tcp-common**

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

**module: ietf-tcp-client**

- 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}?

**module: ietf-tcp-server**

- grouping tcp-server-grouping
  - local-address inet:ip-address
  - local-port? inet:port-number

RFC 8340 - YANG Tree Diagrams

The "common" grouping is factored out to support the possibility of a future "system" grouping.
After flattening all the "uses" statements

module: ietf-tcp-client

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}?
  +-- keepalives! {keepalives-supported}?
    +-- idle-time      uint16 // seconds
    +-- max-probes     uint16
    +-- probe-interval uint16 // seconds

module: ietf-tcp-server

grouping tcp-server-grouping
  +-- local-address  inet:ip-address
  +-- local-port?    inet:port-number
  +-- keepalives! {keepalives-supported}?
    +-- idle-time      uint16 // seconds
    +-- max-probes     uint16
    +-- probe-interval uint16 // seconds
Examples (XML shown, but JSON also used)

```
  <remote-address>www.example.com</remote-address>
  <remote-port>443</remote-port>
  <local-address>0.0.0.0</local-address>  // wildcard value
  <local-port>0</local-port>     // wildcard value
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-client>

<tcp-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-server">
  <local-address>10.20.30.40</local-address>
  <local-port>8888</local-port>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-server>
```
Current Module Dependencies
(IETF-wide)
Next Steps

Honestly, we think that we're done...
- at least for a first release.

But the fundamental question is if this model, as simple as it is, is general and good enough for all situations in which a TCP YANG model could be used?

Questions, Comments, Concerns?