NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

K. Watsen
Juniper Networks

October 22, 2018

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-08

Abstract

This document defines two YANG modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server. Both modules support the TLS transport protocol with both standard RESTCONF and RESTCONF Call Home connections.

Editorial Note (To be removed by RFC Editor)

This draft contains many 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.

This document contains references to other drafts in progress, both in the Normative References section, as well as in body text throughout. Please update the following references to reflect their final RFC assignments:

- I-D.ietf-netconf-keystore
- I-D.ietf-netconf-tls-client-server

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

- "XXXX" --> the assigned RFC value for this draft
- "ZZZZ" --> the assigned RFC value for I-D.ietf-netconf-tls-client-server

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

- "2018-10-22" --> 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/.

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 25, 2019.

Table of Contents

1. Introduction

This document defines two YANG [RFC7950] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server [RFC8040]. Both modules support the TLS [RFC8446] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [RFC8071].

1.1. Terminology

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.

2. The RESTCONF Client Model

The RESTCONF client model presented in this section supports both clients initiating connections to servers, as well as clients listening for connections from servers calling home.

This model, like that presented in [I-D.ietf-netconf-netconf-client-server], is designed to support any number of possible transports. RESTCONF only supports the TLS transport currently, thus this model only supports the TLS transport.

All private keys and trusted certificates are held in the keystore model defined in [I-D.ietf-netconf-keystore].

YANG feature statements are used to enable implementations to advertise which parts of the model the RESTCONF client supports.

2.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-restconf-client" module. Just the container is displayed below, but there is also a reusable grouping called "restconf-client-grouping" that the container is using.

[Note: '\' line wrapping for formatting only]

```
module: ietf-restconf-client
  +--rw restconf-client
     +--rw initiate! {initiate}?
     |  +--rw restconf-server* [name]
     |     +--rw name         string
     |     +--rw endpoints
     |        +--rw endpoint* [name]
     |           +--rw name                  string
     |           +--rw (transport)
     |           |  +--:(tls) {tls-initiate}?
     |           |     +--rw tls
     |           |        +--rw address            inet:host
     |           |        +--rw port?              inet:port-number
     |           |        +--rw client-identity
     |           |        |  +--rw (auth-type)
     |           |        |     +--:(certificate)
     |           |        |        +--rw certificate
     |           |        |           +--rw (local-or-keystore)
     |           |        |              +--:(local)
     |           |        |              |        {local-keys-suppor\
ted}?
     |           |        |              |  +--rw algorithm?
     |           |        |              |  |       asymmetric-key-e\
ncryption-algorithm-ref
     |           |        |              |  +--rw public-key?
     |           |        |              |  |       binary
     |           |        |              |  +--rw private-key?
     |           |        |              |  |       union
     |           |        |              |  +---x generate-hidden-key
     |           |        |              |  |  +---w input
     |           |        |              |  |     +---w algorithm
     |           |        |              |  |             asymmetric\
-key-encryption-algorithm-ref
     |           |        |              |  +---x install-hidden-key
     |           |        |              |  |  +---w input
     |           |        |              |  |     +---w algorithm
     |           |        |              |  |     |       asymmetric\
-key-encryption-algorithm-ref
```

```
       |           |       |             |  |     +---w public-key?
       |           |       |             |  |     |       binary
       |           |       |             |  |     +---w private-key?
       |           |       |             |  |             binary
       |           |       |             |  +--rw cert?
       |           |       |             |  |       end-entity-cert-\
   cms
       |           |       |             |  +---n certificate-expira\
   tion
       |           |       |             |     +-- expiration-date
       |           |       |             |             yang:date-and\
   -time
       |           |       |             +--:(keystore)
       |           |       |                       {keystore-supporte\
   d}?
       |           |       |                  +--rw reference?
       |           |       |                         ks:asymmetric-ke\
   y-certificate-ref
       |           |       +--rw server-auth
       |           |       |  +--rw pinned-ca-certs?
       |           |       |  |       ta:pinned-certificates-ref
       |           |       |  |       {ta:x509-certificates}?
       |           |       |  +--rw pinned-server-certs?
       |           |       |          ta:pinned-certificates-ref
       |           |       |          {ta:x509-certificates}?
       |           |       +--rw hello-params
       |           |               {tls-client-hello-params-config}?
       |           |          +--rw tls-versions
       |           |          |  +--rw tls-version*   identityref
       |           |          +--rw cipher-suites
       |           |             +--rw cipher-suite*   identityref
       |           +--rw connection-type
       |           |  +--rw (connection-type)
       |           |     +--:(persistent-connection)
       |           |     |  +--rw persistent!
       |           |     |     +--rw keep-alives
       |           |     |        +--rw max-wait?       uint16
       |           |     |        +--rw max-attempts?   uint8
       |           |     +--:(periodic-connection)
       |           |        +--rw periodic!
       |           |           +--rw period?         uint16
       |           |           +--rw anchor-time?    yang:date-and-time
       |           |           +--rw idle-timeout?   uint16
       |           +--rw reconnect-strategy
       |              +--rw start-with?     enumeration
       |              +--rw max-attempts?   uint8
       +--rw listen! {listen}?
          +--rw idle-timeout?   uint16
```

```
          +--rw endpoint* [name]
             +--rw name         string
             +--rw (transport)
                +--:(tls) {tls-listen}?
                   +--rw tls
                      +--rw address?          inet:ip-address
                      +--rw port?             inet:port-number
                      +--rw client-identity
                      |  +--rw (auth-type)
                      |     +--:(certificate)
                      |        +--rw certificate
                      |           +--rw (local-or-keystore)
                      |              +--:(local) {local-keys-supported\
   }?
                      |              |  +--rw algorithm?
                      |              |  |       asymmetric-key-encrypt\
   ion-algorithm-ref
                      |              |  +--rw public-key?
                      |              |  |       binary
                      |              |  +--rw private-key?
                      |              |  |       union
                      |              |  +---x generate-hidden-key
                      |              |  |  +---w input
                      |              |  |     +---w algorithm
                      |              |  |             asymmetric-key-e\
   ncryption-algorithm-ref
                      |              |  +---x install-hidden-key
                      |              |  |  +---w input
                      |              |  |     +---w algorithm
                      |              |  |     |       asymmetric-key-e\
   ncryption-algorithm-ref
                      |              |  |     +---w public-key?    bin\
   ary
                      |              |  |     +---w private-key?   bin\
   ary
                      |              |  +--rw cert?
                      |              |  |       end-entity-cert-cms
                      |              |  +---n certificate-expiration
                      |              |     +-- expiration-date
                      |              |             yang:date-and-time
                      |              +--:(keystore) {keystore-supporte\
   d}?
                      |                 +--rw reference?
                      |                         ks:asymmetric-key-cert\
   ificate-ref
                      +--rw server-auth
                      |  +--rw pinned-ca-certs?
                      |  |       ta:pinned-certificates-ref
```

```
                      |  |       {ta:x509-certificates}?
                      |  +--rw pinned-server-certs?
                      |          ta:pinned-certificates-ref
                      |          {ta:x509-certificates}?
                      +--rw hello-params
                             {tls-client-hello-params-config}?
                         +--rw tls-versions
                         |  +--rw tls-version*   identityref
                         +--rw cipher-suites
                            +--rw cipher-suite*   identityref
```

2.2. Example Usage

The following example illustrates configuring a RESTCONF client to initiate connections, as well as listening for call-home connections.

This example is consistent with the examples presented in Section 3.2 of [I-D.ietf-netconf-keystore].

[Note: '\' line wrapping for formatting only]

```
<restconf-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-client">

  <!-- RESTCONF servers to initiate connections to -->
  <initiate>
    <restconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <tls>
            <address>corp-fw1.example.com</address>
            <client-identity>
              <certificate>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
f-crypto-types">ct:rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
                <cert>base64encodedvalue==</cert>
              </certificate>
            </client-identity>
            <server-auth>
              <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
inned-ca-certs>
              <pinned-server-certs>explicitly-trusted-server-certs</\
pinned-server-certs>
```

```
              </server-auth>
            </tls>
            <connection-type>
              <persistent/>
            </connection-type>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <tls>
              <address>corp-fw2.example.com</address>
              <client-identity>
                <certificate>
                  <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
   f-crypto-types">ct:rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                  <cert>base64encodedvalue==</cert>
                </certificate>
              </client-identity>
              <server-auth>
                <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
   inned-ca-certs>
                <pinned-server-certs>explicitly-trusted-server-certs</\
   pinned-server-certs>
              </server-auth>
            </tls>
            <connection-type>
              <persistent/>
            </connection-type>
          </endpoint>
        </endpoints>
      </restconf-server>
    </initiate>

    <!-- endpoints to listen for RESTCONF Call Home connections on -->
    <listen>
      <endpoint>
        <name>Intranet-facing listener</name>
        <tls>
          <address>11.22.33.44</address>
          <client-identity>
            <certificate>
              <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cr\
   ypto-types">ct:rsa2048</algorithm>
              <private-key>base64encodedvalue==</private-key>
              <public-key>base64encodedvalue==</public-key>
              <cert>base64encodedvalue==</cert>
            </certificate>
```

```
          </client-identity>
          <server-auth>
            <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinne\
d-ca-certs>
            <pinned-server-certs>explicitly-trusted-server-certs</pinn\
ed-server-certs>
          </server-auth>
        </tls>
      </endpoint>
    </listen>
  </restconf-client>
```

2.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC8040], and [RFC8071], and [I-D.ietf-netconf-tls-client-server].

```
<CODE BEGINS> file "ietf-restconf-client@2018-10-22.yang"
module ietf-restconf-client {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-client";
  prefix "rcc";

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

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

  import ietf-tls-client {
    prefix ts;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
  }

  organization
   "IETF NETCONF (Network Configuration) Working Group";

  contact
   "WG Web:   <http://datatracker.ietf.org/wg/restconf/>
```

```
       WG List:  <mailto:restconf@ietf.org>

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>

       Author:   Gary Wu
                 <mailto:garywu@cisco.com>";


     description
      "This module contains a collection of YANG definitions for
       configuring RESTCONF clients.

       Copyright (c) 2017 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.";

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: RESTCONF Client and Server Models";
     }

     // Features

     feature initiate {
       description
        "The 'initiate' feature indicates that the RESTCONF client
         supports initiating RESTCONF connections to RESTCONF servers
         using at least one transport (e.g., TLS, etc.).";
     }

     feature tls-initiate {
       if-feature initiate;
       description
        "The 'tls-initiate' feature indicates that the RESTCONF client
         supports initiating TLS connections to RESTCONF servers. This
         feature exists as TLS might not be a mandatory to implement
```

```
         transport in the future.";
       reference
        "RFC 8040: RESTCONF Protocol";
     }

     feature listen {
       description
        "The 'listen' feature indicates that the RESTCONF client
         supports opening a port to accept RESTCONF server call
         home connections using at least one transport (e.g.,
         TLS, etc.).";
     }

     feature tls-listen {
       if-feature listen;
       description
        "The 'tls-listen' feature indicates that the RESTCONF client
         supports opening a port to listen for incoming RESTCONF
         server call-home TLS connections.  This feature exists as
         TLS might not be a mandatory to implement transport in the
         future.";
       reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
     }

     container restconf-client {
       uses restconf-client-grouping;
       description
         "Top-level container for RESTCONF client configuration.";
     }

     grouping restconf-client-grouping {
       description
         "Top-level grouping for RESTCONF client configuration.";

       container initiate {
         if-feature initiate;
         presence "Enables client to initiate TCP connections";
         description
           "Configures client initiating underlying TCP connections.";
         list restconf-server {
           key name;
           min-elements 1;
           description
             "List of RESTCONF servers the RESTCONF client is to
              initiate connections to in parallel.";
           leaf name {
             type string;
```

```
            description
              "An arbitrary name for the RESTCONF server.";
          }
          container endpoints {
            description
              "Container for the list of endpoints.";
            list endpoint {
              key name;
              min-elements 1;
              ordered-by user;
              description
                "A non-empty user-ordered list of endpoints for this
                 RESTCONF client to try to connect to in sequence.
                 Defining more than one enables high-availability.";
              leaf name {
                type string;
                description
                  "An arbitrary name for this endpoint.";
              }
              choice transport {
                mandatory true;
                description
                  "Selects between available transports. This is a
                   'choice' statement so as to support additional
                   transport options to be augmented in.";
                case tls {
                  if-feature tls-initiate;
                  container tls {
                    description
                      "Specifies TLS-specific transport
                       configuration.";
                    leaf address {
                      type inet:host;
                      mandatory true;
                      description
                       "The IP address or hostname of the endpoint.
                        If a domain name is configured, then the
                        DNS resolution should happen on each usage
                        attempt.  If the the DNS resolution results
                        in multiple IP addresses, the IP addresses
                        will be tried according to local preference
                        order until a connection has been established
                        or until all IP addresses have failed.";
                    }
                    leaf port {
                      type inet:port-number;
                      default 443;
                      description
```

```
                    "The IP port for this endpoint. The RESTCONF
                     client will use the IANA-assigned well-known
                     port for 'https' (443) if no value is
                     specified.";
                  }
                  uses ts:tls-client-grouping {
                    refine "client-identity/auth-type" {
                      mandatory true;
                      description
                        "RESTCONF clients MUST pass some
                         authentication credentials.";
                    }
                  }
                }
              } // end tls
            } // end transport
            container connection-type {
              description
               "Indicates the kind of connection to use.";
              choice connection-type {
                mandatory true;
                description
                  "Selects between available connection types.";
                case persistent-connection {
                  container persistent {
                    presence
                     "Indicates that a persistent connection is
                      to be maintained.";
                    description
                     "Maintain a persistent connection to the
                      RESTCONF server. If the connection goes down,
                      immediately start trying to reconnect to it,
                      using the reconnection strategy.  This
                      connection type minimizes any RESTCONF server
                      to RESTCONF client data-transfer delay, albeit
                      at the expense of holding resources longer.";
                    container keep-alives {
                      description
                        "Configures the keep-alive policy, to
                         proactively test the aliveness of the TLS
                         server.  An unresponsive TLS server will
                         be dropped after approximately max-attempts
                         * max-wait seconds.";
                      leaf max-wait {
                        type uint16 {
                          range "1..max";
                        }
                        units seconds;
```

```
            default 30;
            description
             "Sets the amount of time in seconds after
              which if no data has been received from
              the TLS server, a TLS-level message will
              be sent to test the aliveness of the TLS
              server.";
          }
          leaf max-attempts {
            type uint8;
            default 3;
            description
             "Sets the maximum number of sequential
              keep-alive messages that can fail to
              obtain a response from the TLS server
              before assuming the TLS server is no
              longer alive.";
          }
        }
      }
    }
    case periodic-connection {
      container periodic {
        presence
         "Indicates that a periodic connection is to be
          maintained.";
        description
         "Periodically connect to the NETCONF server.
          The RESTCONF server should close the underlying
          TLS connection upon completing planned
          activities.

          This connection type increases resource
          utilization, albeit with increased delay in
          RESTCONF server to RESTCONF client
          interactions.";
        leaf period {
          type uint16;
          units "minutes";
          default 60;
          description
            "Duration of time between periodic
             connections.";
        }
        leaf anchor-time {
          type yang:date-and-time {
            // constrained to minute-level granularity
            pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
```

```
                    + '(Z|[\+\-]\d{2}:\d{2})';
                }
                description
                  "Designates a timestamp before or after which
                   a series of periodic connections are
                   determined.  The periodic connections occur
                   at a whole multiple interval from the anchor
                   time.  For example, for an anchor time is 15
                   minutes past midnight and a period interval
                   of 24 hours, then a periodic connection will
                   occur 15 minutes past midnight everyday.";
              }
              leaf idle-timeout {
                type uint16;
                units "seconds";
                default 120; // two minutes
                description
                  "Specifies the maximum number of seconds
                   that the underlying TLS session may remain
                   idle. A TLS session will be dropped if it
                   is idle for an interval longer than this
                   number of seconds If set to zero, then the
                   RESTCONF client will never drop a session
                   because it is idle.";
              }
            }
          } // end periodic-connection
        } // end connection-type
      } // end connection-type
      container reconnect-strategy {
        description
         "The reconnection strategy directs how a RESTCONF
          client reconnects to a RESTCONF server, after
          discovering its connection to the server has
          dropped, even if due to a reboot.  The RESTCONF
          client starts with the specified endpoint and
          tries to connect to it max-attempts times before
          trying the next endpoint in the list (round
          robin).";
        leaf start-with {
          type enumeration {
            enum first-listed {
              description
                "Indicates that reconnections should start
                 with the first endpoint listed.";
            }
            enum last-connected {
              description
```

```
                    "Indicates that reconnections should start
                     with the endpoint last connected to.  If
                     no previous connection has ever been
                     established, then the first endpoint
                     configured is used.   RESTCONF clients
                     SHOULD be able to remember the last
                     endpoint connected to across reboots.";
                  }
                  enum random-selection {
                    description
                      "Indicates that reconnections should start with
                       a random endpoint.";
                  }
                }
                default first-listed;
                description
                 "Specifies which of the RESTCONF server's
                  endpoints the RESTCONF client should start
                  with when trying to connect to the RESTCONF
                  server.";
              }
              leaf max-attempts {
                type uint8 {
                  range "1..max";
                }
                default 3;
                description
                 "Specifies the number times the RESTCONF client
                  tries to connect to a specific endpoint before
                  moving on to the next endpoint in the list
                  (round robin).";
              }
            } // end reconnect-strategy
          } // end endpoint
        } // end endpoints
      } // end restconf-server
    } // end initiate

    container listen {
      if-feature listen;
      presence "Enables client to accept call-home connections";
      description
        "Configures client accepting call-home TCP connections.";

      leaf idle-timeout {
        type uint16;
        units "seconds";
        default 3600; // one hour
```

```
            description
              "Specifies the maximum number of seconds that an
               underlying TLS session may remain idle. A TLS session
               will be dropped if it is idle for an interval longer
               than this number of seconds.  If set to zero, then
               the server will never drop a session because it is
               idle.  Sessions that have a notification subscription
               active are never dropped.";
          }

          list endpoint {
            key name;
            min-elements 1;
            description
              "List of endpoints to listen for RESTCONF connections.";
            leaf name {
              type string;
              description
                "An arbitrary name for the RESTCONF listen endpoint.";
            }
            choice transport {
              mandatory true;
              description
                "Selects between available transports. This is a
                 'choice' statement so as to support additional
                 transport options to be augmented in.";
              case tls {
                if-feature tls-listen;
                container tls {
                  description
                    "TLS-specific listening configuration for inbound
                     connections.";
                  leaf address {
                    type inet:ip-address;
                    description
                     "The IP address to listen on for incoming call-
                      home connections.  The RESTCONF client will
                      listen on all configured interfaces if no
                      value is specified.  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 port {
                    type inet:port-number;
                    default 4336;
                    description
                     "The port number to listen on for call-home
```

```
                    connections. The RESTCONF client will listen
                    on the IANA-assigned well-known port for
                    'restconf-ch-tls' (4336) if no value is
                     specified.";
                }
                uses ts:tls-client-grouping {
                  refine "client-identity/auth-type" {
                    mandatory true;
                    description
                      "RESTCONF clients MUST pass some authentication
                       credentials.";
                  }
                }
              }
            }
          } // end transport
        } // end endpoint
      } // end listen
    } // end restconf-client
  }
  <CODE ENDS>
```

3. The RESTCONF Server Model

The RESTCONF server model presented in this section supports servers both listening for connections as well as initiating call-home connections.

All private keys and trusted certificates are held in the keystore model defined in [I-D.ietf-netconf-keystore].

YANG feature statements are used to enable implementations to advertise which parts of the model the RESTCONF server supports.

3.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-restconf-server" module. Just the container is displayed below, but there is also a reusable grouping called "restconf-server-grouping" that the container is using.

[Note: '\' line wrapping for formatting only]

```
module: ietf-restconf-server
  +--rw restconf-server
     +--rw listen! {listen}?
     |  +--rw endpoint* [name]
```

```
        |     +--rw name         string
        |     +--rw (transport)
        |        +--:(tls) {tls-listen}?
        |           +--rw tls
        |              +--rw address?          inet:ip-address
        |              +--rw port?             inet:port-number
        |              +--rw server-identity
        |              |  +--rw (local-or-keystore)
        |              |     +--:(local) {local-keys-supported}?
        |              |     |  +--rw algorithm?
        |              |     |  |       asymmetric-key-encryption-algor\
   ithm-ref
        |              |     |  +--rw public-key?              binary
        |              |     |  +--rw private-key?             union
        |              |     |  +---x generate-hidden-key
        |              |     |  |  +---w input
        |              |     |  |     +---w algorithm
        |              |     |  |             asymmetric-key-encryption\
   -algorithm-ref
        |              |     |  +---x install-hidden-key
        |              |     |  |  +---w input
        |              |     |  |     +---w algorithm
        |              |     |  |     |       asymmetric-key-encryption\
   -algorithm-ref
        |              |     |  |     +---w public-key?    binary
        |              |     |  |     +---w private-key?   binary
        |              |     |  +--rw cert?
        |              |     |  |       end-entity-cert-cms
        |              |     |  +---n certificate-expiration
        |              |     |     +-- expiration-date
        |              |     |             yang:date-and-time
        |              |     +--:(keystore) {keystore-supported}?
        |              |        +--rw reference?
        |              |                ks:asymmetric-key-certificate-r\
   ef
        |              +--rw client-auth
        |              |  +--rw pinned-ca-certs?
        |              |  |       ta:pinned-certificates-ref
        |              |  |       {ta:x509-certificates}?
        |              |  +--rw pinned-client-certs?
        |              |  |       ta:pinned-certificates-ref
        |              |  |       {ta:x509-certificates}?
        |              |  +--rw cert-maps
        |              |     +--rw cert-to-name* [id]
        |              |        +--rw id             uint32
        |              |        +--rw fingerprint
        |              |        |       x509c2n:tls-fingerprint
        |              |        +--rw map-type       identityref
```

```
        |              |        +--rw name           string
        |              +--rw hello-params
        |                       {tls-server-hello-params-config}?
        |                 +--rw tls-versions
        |                 |  +--rw tls-version*   identityref
        |                 +--rw cipher-suites
        |                    +--rw cipher-suite*   identityref
        +--rw call-home! {call-home}?
           +--rw restconf-client* [name]
              +--rw name                  string
              +--rw endpoints
              |  +--rw endpoint* [name]
              |     +--rw name         string
              |     +--rw (transport)
              |        +--:(tls) {tls-call-home}?
              |           +--rw tls
              |              +--rw address            inet:host
              |              +--rw port?              inet:port-number
              |              +--rw server-identity
              |              |  +--rw (local-or-keystore)
              |              |     +--:(local) {local-keys-supported}?
              |              |     |  +--rw algorithm?
              |              |     |  |       asymmetric-key-encryption\
   -algorithm-ref
              |              |     |  +--rw public-key?
              |              |     |  |       binary
              |              |     |  +--rw private-key?
              |              |     |  |       union
              |              |     |  +---x generate-hidden-key
              |              |     |  |  +---w input
              |              |     |  |     +---w algorithm
              |              |     |  |             asymmetric-key-encr\
   yption-algorithm-ref
              |              |     |  +---x install-hidden-key
              |              |     |  |  +---w input
              |              |     |  |     +---w algorithm
              |              |     |  |     |       asymmetric-key-encr\
   yption-algorithm-ref
              |              |     |  |     +---w public-key?    binary
              |              |     |  |     +---w private-key?   binary
              |              |     |  +--rw cert?
              |              |     |  |       end-entity-cert-cms
              |              |     |  +---n certificate-expiration
              |              |     |     +-- expiration-date
              |              |     |             yang:date-and-time
              |              |     +--:(keystore) {keystore-supported}?
              |              |        +--rw reference?
              |              |                ks:asymmetric-key-certifi\
```

```
   cate-ref
               |              +--rw client-auth
               |              |  +--rw pinned-ca-certs?
               |              |  |       ta:pinned-certificates-ref
               |              |  |       {ta:x509-certificates}?
               |              |  +--rw pinned-client-certs?
               |              |  |       ta:pinned-certificates-ref
               |              |  |       {ta:x509-certificates}?
               |              |  +--rw cert-maps
               |              |     +--rw cert-to-name* [id]
               |              |        +--rw id             uint32
               |              |        +--rw fingerprint
               |              |        |       x509c2n:tls-fingerprint
               |              |        +--rw map-type       identityref
               |              |        +--rw name           string
               |              +--rw hello-params
               |                      {tls-server-hello-params-config}?
               |                 +--rw tls-versions
               |                 |  +--rw tls-version*   identityref
               |                 +--rw cipher-suites
               |                    +--rw cipher-suite*   identityref
               |           +--rw connection-type
               |           |  +--rw (connection-type)
               |           |     +--:(persistent-connection)
               |           |     |  +--rw persistent!
               |           |     |     +--rw keep-alives
               |           |     |        +--rw max-wait?       uint16
               |           |     |        +--rw max-attempts?   uint8
               |           |     +--:(periodic-connection)
               |           |        +--rw periodic!
               |           |           +--rw period?         uint16
               |           |           +--rw anchor-time?    yang:date-and-time
               |           |           +--rw idle-timeout?   uint16
               |           +--rw reconnect-strategy
               |              +--rw start-with?     enumeration
               |              +--rw max-attempts?   uint8
```

3.2. Example Usage

The following example illustrates configuring a RESTCONF server to listen for RESTCONF client connections, as well as configuring call-home to one RESTCONF client.

This example is consistent with the examples presented in Section 3.2 of [I-D.ietf-netconf-keystore].

[Note: '\' line wrapping for formatting only]

```
   <restconf-server
     xmlns="urn:ietf:params:xml:ns:yang:ietf-restconf-server"
     xmlns:x509c2n="urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">

     <!-- endpoints to listen for RESTCONF connections on -->
     <listen>
       <endpoint>
         <name>netconf/tls</name>
         <tls>
           <address>11.22.33.44</address>
           <server-identity>
             <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cryp\
   to-types">ct:rsa2048</algorithm>
             <private-key>base64encodedvalue==</private-key>
             <public-key>base64encodedvalue==</public-key>
             <cert>base64encodedvalue==</cert>
           </server-identity>
           <client-auth>
             <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinne\
   d-ca-certs>
             <pinned-client-certs>explicitly-trusted-client-certs</pinn\
   ed-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>B3:4F:A1:8C:54</fingerprint>
                 <map-type>x509c2n:specified</map-type>
                 <name>scooby-doo</name>
               </cert-to-name>
             </cert-maps>
           </client-auth>
         </tls>
       </endpoint>
     </listen>

     <!-- call home to a RESTCONF client with two endpoints -->
     <call-home>
       <restconf-client>
         <name>config-manager</name>
         <endpoints>
           <endpoint>
             <name>east-data-center</name>
             <tls>
```

```
              <address>22.33.44.55</address>
              <server-identity>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
   crypto-types">ct:rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
                <cert>base64encodedvalue==</cert>
              </server-identity>
              <client-auth>
                <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
   inned-ca-certs>
                <pinned-client-certs>explicitly-trusted-client-certs</\
   pinned-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>B3:4F:A1:8C:54</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                </cert-maps>
              </client-auth>
            </tls>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <tls>
              <address>33.44.55.66</address>
              <server-identity>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
   crypto-types">ct:rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
                <cert>base64encodedvalue==</cert>
              </server-identity>
              <client-auth>
                <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
   inned-ca-certs>
                <pinned-client-certs>explicitly-trusted-client-certs</\
   pinned-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>B3:4F:A1:8C:54</fingerprint>
                    <map-type>x509c2n:specified</map-type>
                    <name>scooby-doo</name>
                  </cert-to-name>
                </cert-maps>
              </client-auth>
            </tls>
          </endpoint>
        </endpoints>
        <connection-type>
          <periodic>
            <idle-timeout>300</idle-timeout>
            <period>60</period>
          </periodic>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </restconf-client>
    </call-home>
  </restconf-server>
```

3.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC7407], [RFC8040], [RFC8071], and [I-D.ietf-netconf-tls-client-server].

```
<CODE BEGINS> file "ietf-restconf-server@2018-10-22.yang"
module ietf-restconf-server {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-server";
  prefix "rcs";

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

  import ietf-inet-types {
```

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

  import ietf-x509-cert-to-name {
    prefix x509c2n;
    reference
      "RFC 7407: A YANG Data Model for SNMP Configuration";
  }

  import ietf-tls-server {
    prefix ts;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
  }

  organization
   "IETF NETCONF (Network Configuration) Working Group";

  contact
   "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>

    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>

    Author:   Gary Wu
              <mailto:garywu@cisco.com>

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


  description
   "This module contains a collection of YANG definitions for
    configuring RESTCONF servers.

    Copyright (c) 2017 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.";

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: RESTCONF Client and Server Models";
     }


     // Features

     feature listen {
       description
        "The 'listen' feature indicates that the RESTCONF server
         supports opening a port to accept RESTCONF client connections
         using at least one transport (e.g., TLS, etc.).";
     }

     feature tls-listen {
       if-feature listen;
       description
        "The 'tls-listen' feature indicates that the RESTCONF server
         supports opening a port to listen for incoming RESTCONF
         client connections.  This feature exists as TLS might not
         be a mandatory to implement transport in the future.";
       reference
        "RFC 8040: RESTCONF Protocol";
     }

     feature call-home {
       description
        "The 'call-home' feature indicates that the RESTCONF
         server supports initiating RESTCONF call home connections
         to RESTCONF clients using at least one transport (e.g.,
         TLS, etc.).";
       reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
     }

     feature tls-call-home {
       if-feature call-home;
       description
        "The 'tls-call-home' feature indicates that the RESTCONF
         server supports initiating connections to RESTCONF clients.
         This feature exists as TLS might not be a mandatory to
         implement transport in the future.";
```

```
      reference
       "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

    container restconf-server {
      uses restconf-server-grouping;
      description
        "Top-level container for RESTCONF server configuration.";
    }

    grouping restconf-server-grouping {
      description
        "Top-level grouping for RESTCONF server configuration.";

      container listen {
        if-feature listen;
        presence "Enables server to listen for TCP connections";
        description "Configures listen behavior";
        list endpoint {
          key name;
          min-elements 1;
          description
            "List of endpoints to listen for RESTCONF connections.";
          leaf name {
            type string;
            description
              "An arbitrary name for the RESTCONF listen endpoint.";
          }
          choice transport {
            mandatory true;
            description
              "Selects between available transports. This is a
               'choice' statement so as to support additional
               transport options to be augmented in.";
            case tls {
              if-feature tls-listen;
              container tls {
                description
                  "TLS-specific listening configuration for inbound
                   connections.";
                leaf address {
                  type inet:ip-address;
                description
                  "The IP address to listen on for incoming
                   connections.  The RESTCONF server will listen
                   on all configured interfaces if no value is
                   specified.  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 port {
                  type inet:port-number;
                  default 443;
                  description
                   "The local port number to listen on.  If no value
                    is specified, the IANA-assigned port value for
                    'https' (443) is used.";
                }
                uses ts:tls-server-grouping {
                  refine "client-auth" {
                    must 'pinned-ca-certs or pinned-client-certs';
                    description
                      "RESTCONF servers MUST be able to validate
                       clients.";
                  }
                  augment "client-auth" {
                    description
                      "Augments in the cert-to-name structure,
                       so the RESTCONF server can map TLS-layer
                       client certificates to RESTCONF usernames.";
                    container cert-maps {
                      uses x509c2n:cert-to-name;
                      description
                       "The cert-maps container is used by a TLS-
                        based RESTCONF server to map the RESTCONF
                        client's presented X.509 certificate to
                        a RESTCONF username.  If no matching and
                        valid cert-to-name list entry can be found,
                        then the RESTCONF server MUST close the
                        connection, and MUST NOT accept RESTCONF
                        messages over it.";
                      reference
                        "RFC 7407: A YANG Data Model for SNMP
                                   Configuration.";
                    }
                  }
                }
              } // end tls container
            } // end tls case
          } // end transport
        } // end endpoint
      } // end listen

      container call-home {
        if-feature call-home;
```

```
          presence "Enables server to initiate TCP connections";
          description "Configures call-home behavior";
          list restconf-client {
            key name;
            min-elements 1;
            description
              "List of RESTCONF clients the RESTCONF server is to
               initiate call-home connections to in parallel.";
            leaf name {
              type string;
              description
                "An arbitrary name for the remote RESTCONF client.";
            }
            container endpoints {
              description
                "Container for the list of endpoints.";
              list endpoint {
                key name;
                min-elements 1;
                ordered-by user;
                description
                  "User-ordered list of endpoints for this RESTCONF
                   client.  Defining more than one enables high-
                   availability.";
                leaf name {
                  type string;
                  description
                    "An arbitrary name for this endpoint.";
                }
                choice transport {
                  mandatory true;
                  description
                    "Selects between available transports. This is a
                     'choice' statement so as to support additional
                     transport options to be augmented in.";
                  case tls {
                    if-feature tls-call-home;
                    container tls {
                      description
                        "Specifies TLS-specific call-home transport
                         configuration.";
                      leaf address {
                        type inet:host;
                        mandatory true;
                        description
                         "The IP address or hostname of the endpoint.
                          If a domain name is configured, then the
                          DNS resolution should happen on each usage
```

```
                    attempt.  If the DNS resolution results in
                    multiple IP addresses, the IP addresses will
                    be tried according to local preference order
                    until a connection has been established or
                    until all IP addresses have failed.";
                }
                leaf port {
                  type inet:port-number;
                  default 4336;
                  description
                   "The IP port for this endpoint.  The RESTCONF
                    server will use the IANA-assigned well-known
                    port for 'restconf-ch-tls' (4336) if no value
                    is specified.";
                }
                uses ts:tls-server-grouping {
                  refine "client-auth" {
                   must 'pinned-ca-certs or pinned-client-certs';
                   description
                    "RESTCONF servers MUST be able to validate
                     clients.";
                  }
                  augment "client-auth" {
                    description
                     "Augments in the cert-to-name structure,
                      so the RESTCONF server can map TLS-layer
                      client certificates to RESTCONF usernames.";
                    container cert-maps {
                      uses x509c2n:cert-to-name;
                      description
                       "The cert-maps container is used by a
                        TLS-based RESTCONF server to map the
                        RESTCONF client's presented X.509
                        certificate to a RESTCONF username. If
                        no matching and valid cert-to-name list
                        entry can be found, then the RESTCONF
                        server MUST close the connection, and
                        MUST NOT accept RESTCONF messages over
                        it.";
                      reference
                        "RFC 7407: A YANG Data Model for SNMP
                         Configuration.";
                    }
                  }
                }
              }
            }
          } // end transport
```

```
        } // end endpoint
      } // end endpoints
      container connection-type {
        description
         "Indicates the RESTCONF client's preference for how the
          RESTCONF server's connection is maintained.";
        choice connection-type {
          mandatory true;
          description
            "Selects between available connection types.";
          case persistent-connection {
            container persistent {
              presence
               "Indicates that a persistent connection is to be
                maintained.";
              description
               "Maintain a persistent connection to the RESTCONF
                client. If the connection goes down, immediately
                start trying to reconnect to it, using the
                reconnection strategy.

                This connection type minimizes any RESTCONF
                client to RESTCONF server data-transfer delay,
                albeit at the expense of holding resources
                longer.";
              container keep-alives {
                description
                  "Configures the keep-alive policy, to
                   proactively test the aliveness of the TLS
                   client.  An unresponsive TLS client will
                   be dropped after approximately (max-attempts
                   * max-wait) seconds.";
                reference
                  "RFC 8071: NETCONF Call Home and RESTCONF
                              Call Home, Section 4.1, item S7";
                leaf max-wait {
                  type uint16 {
                    range "1..max";
                  }
                  units seconds;
                  default 30;
                  description
                   "Sets the amount of time in seconds after
                    which if no data has been received from
                    the TLS client, a TLS-level message will
                    be sent to test the aliveness of the TLS
                    client.";
                }
```

```
                leaf max-attempts {
                  type uint8;
                  default 3;
                  description
                   "Sets the maximum number of sequential keep-
                    alive messages that can fail to obtain a
                    response from the TLS client before assuming
                    the TLS client is no longer alive.";
                }
              }
            }
          }
          case periodic-connection {
            container periodic {
              presence
               "Indicates that a periodic connection is to be
                maintained.";
              description
               "Periodically connect to the RESTCONF client.  The
                RESTCONF client should close the underlying TLS
                connection upon completing planned activities.

                This connection type increases resource
                utilization, albeit with increased delay in
                RESTCONF client to RESTCONF client interactions.";
              leaf period {
                type uint16;
                units "minutes";
                default 60;
                description
                  "Duration of time between periodic connections.";
              }
              leaf anchor-time {
                type yang:date-and-time {
                  // constrained to minute-level granularity
                  pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                          + '(Z|[\+\-]\d{2}:\d{2})';
                }
                description
                  "Designates a timestamp before or after which a
                   series of periodic connections are determined.
                   The periodic connections occur at a whole
                   multiple interval from the anchor time.  For
                   example, for an anchor time is 15 minutes past
                   midnight and a period interval of 24 hours, then
                   a periodic connection will occur 15 minutes past
                   midnight everyday.";
              }
```

```
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default 120; // two minutes
                  description
                    "Specifies the maximum number of seconds that
                     the underlying TLS session may remain idle.
                     A TLS session will be dropped if it is idle
                     for an interval longer than this number of
                     seconds.  If set to zero, then the server
                     will never drop a session because it is idle.";
                }
              }
            }
          }
        }
        container reconnect-strategy {
          description
           "The reconnection strategy directs how a RESTCONF server
            reconnects to a RESTCONF client after discovering its
            connection to the client has dropped, even if due to a
            reboot.  The RESTCONF server starts with the specified
            endpoint and tries to connect to it max-attempts times
            before trying the next endpoint in the list (round
            robin).";
          leaf start-with {
            type enumeration {
              enum first-listed {
                description
                  "Indicates that reconnections should start with
                   the first endpoint listed.";
              }
              enum last-connected {
                description
                  "Indicates that reconnections should start with
                   the endpoint last connected to.  If no previous
                   connection has ever been established, then the
                   first endpoint configured is used.   RESTCONF
                   servers SHOULD be able to remember the last
                   endpoint connected to across reboots.";
              }
              enum random-selection {
                description
                  "Indicates that reconnections should start with
                   a random endpoint.";
              }
            }
            default first-listed;
```

```
              description
               "Specifies which of the RESTCONF client's endpoints
                the RESTCONF server should start with when trying
                to connect to the RESTCONF client.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default 3;
              description
               "Specifies the number times the RESTCONF server tries
                to connect to a specific endpoint before moving on to
                the next endpoint in the list (round robin).";
            }
          }
        }
      }
    }
  }
<CODE ENDS>
```

4. Security Considerations

The YANG module defined in this document uses a grouping defined in [I-D.ietf-netconf-tls-client-server]. Please see the Security Considerations section in that document for concerns related that grouping.

The YANG module defined in this document is 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.

There are a number of data nodes defined in this YANG module that 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 operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/: The entire data trees defined by the modules defined in this draft are sensitive to write operations. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., can dramatically alter the implemented security policy. However, no NACM annotations are applied as the data SHOULD be editable by users other than a designated 'recovery session'.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

NONE

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

NONE

5. IANA Considerations

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

5.2. The YANG Module Names Registry

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

```
      name:         ietf-restconf-client
      namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-client
      prefix:       ncc
      reference:    RFC XXXX

      name:         ietf-restconf-server
      namespace:    urn:ietf:params:xml:ns:yang:ietf-restconf-server
      prefix:       ncs
      reference:    RFC XXXX
```

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-06 (work in progress), September 2018.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-07 (work in progress), September 2018.

[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>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[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>.

6.2. 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.

[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>.

[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>.

[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>.

[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>.

Appendix A. Change Log

A.1. 00 to 01

- Renamed "keychain" to "keystore".

A.2. 01 to 02

- Filled in previously missing 'ietf-restconf-client' module.
- Updated the ietf-restconf-server module to accomodate new grouping 'ietf-tls-server-grouping'.

A.3. 02 to 03

- Refined use of tls-client-grouping to add a must statement indicating that the TLS client must specify a client-certificate.
- Changed restconf-client??? to be a grouping (not a container).

A.4. 03 to 04

- Added RFC 8174 to Requirements Language Section.
- Replaced refine statement in ietf-restconf-client to add a mandatory true.
- Added refine statement in ietf-restconf-server to add a must statement.
- Now there are containers and groupings, for both the client and server models.
- Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- Updated examples to inline key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

- Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- Updated examples to inline key and certificates (no longer a leafref to keystore)

A.6. 05 to 06

- Fixed change log missing section issue.
- Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.
- Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

- removed "idle-timeout" from "persistent" connection config.
- Added "random-selection" for reconnection-strategy's "starts-with" enum.
- Replaced "connection-type" choice default (persistent) with "mandatory true".
- Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.
- Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

- Modified examples to be compatible with new crypto-types algs

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, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, and Bert Wijnen.

Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net