NETCONF Working Group K. Watsen
Internet-Draft Juniper Networks
Intended status: Standards Track J. Schoenwaelder
Expires: January 4, 2018 Jacobs University Bremen
July 3, 2017

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-04

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:

- "2017-07-03" --> the publication date of this draft

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

- o Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Table of Contents

1. Introduction 3
   1.1. Terminology 3
   1.2. Tree Diagrams 3
2. The RESTCONF Client Model 4
   2.1. Tree Diagram 4
   2.2. Example Usage 6
   2.3. YANG Model 8
3. The RESTCONF Server Model 16
   3.1. Tree Diagram 17
   3.2. Example Usage 18
   3.3. YANG Model 20
4. Security Considerations 29

5. IANA Considerations 30
5.1. The IETF XML Registry 30
5.2. The YANG Module Names Registry 31
6. Acknowledgements 31
7. References 31
7.1. Normative References 31
7.2. Informative References 32
Appendix A. Change Log 34
A.1. server-model-09 to 00 34
A.2. 00 to 01 34
A.3. 01 to 02 34
A.4. 02 to 03 34
A.5. 03 to 04 34
Authors' Addresses 34

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 [RFC5246] 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.

1.2. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.

- Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.

- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. The RESTCONF Client Model

EDITOR NOTE: Please ignore this section, it is incomplete.

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 supports the TLS transport protocol using the TLS client groupings defined in [I-D.ietf-netconf-tls-client-server].

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

Just the container is displayed below, but there is also a grouping that the container is using.

Note: all lines are folded at column 71 with no '\' character.

```
module: ietf-restconf-client
    +--rw restconf-client
       +--rw initiate {initiate}?
       |  +--rw restconf-server* [name]
       |     +--rw name                  string
       |     +--rw (transport)
       |     |  +--:(tls) {tls-initiate}?
       |     |     +--rw tls
       |     |        +--rw endpoints
       |     |        |  +--rw endpoint* [name]
       |     |        |     +--rw name       string
       |     |        |     +--rw address    inet:host
       |     |        |     +--rw port?      inet:port-number
       |     |        +--rw server-auth
       |     |        |  +--rw trusted-ca-certs?       leafref
       |     |        |  +--rw trusted-server-certs?   leafref
       |     |        +--rw client-auth
       |     |        |  +--rw (auth-type)
```

```
          |     |        |     +--:(certificate)
          |     |        |        +--rw certificate?   leafref
          |     |        +--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 idle-timeout?   uint32
          |     |     |     +--rw keep-alives
          |     |     |        +--rw max-wait?       uint16
          |     |     |        +--rw max-attempts?   uint8
          |     |     +--:(periodic-connection)
          |     |        +--rw periodic!
          |     |           +--rw idle-timeout?        uint16
          |     |           +--rw reconnect-timeout?   uint16
          |     +--rw reconnect-strategy
          |        +--rw start-with?     enumeration
          |        +--rw max-attempts?   uint8
          +--rw listen {listen}?
             +--rw max-sessions?   uint16
             +--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 server-auth
                         |  +--rw trusted-ca-certs?       leafref
                         |  +--rw trusted-server-certs?   leafref
                         +--rw client-auth
                         |  +--rw (auth-type)
                         |     +--:(certificate)
                         |        +--rw certificate?   leafref
                         +--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 2.2 of [I-D.ietf-netconf-keystore].

```
<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>
      <tls>
        <endpoints>
          <endpoint>
            <name>corp-fw1.example.com</name>
            <address>corp-fw1.example.com</address>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <address>corp-fw2.example.com</address>
          </endpoint>
        </endpoints>
        <server-auth>
          <trusted-server-certs>deployment-specific-ca-certs</trusted-server-
certs>
        </server-auth>
        <client-auth>
          <certificate>tls-ec-cert</certificate>
        </client-auth>
      </tls>
    </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>
        <server-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-server-certs>explicitly-trusted-server-certs</trusted-
server-certs>
        </server-auth>
        <client-auth>
          <certificate>tls-ec-cert</certificate>
        </client-auth>
      </tls>
    </endpoint>
  </listen>
</restconf-client>
```

2.3. YANG Model

This YANG module imports YANG types from [RFC6991] and [RFC7407].

```
<CODE BEGINS> file "ietf-restconf-client@2017-07-03.yang"

module ietf-restconf-client {
  yang-version 1.1;

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

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

  import ietf-tls-client {
    prefix ts;
    revision-date 2017-06-13; // stable grouping definitions
    reference
      "RFC ZZZZ: TLS Client and Server Models";
  }

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

  contact
   "WG Web:   <http://tools.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) 2014 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
```

```
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD
       License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

     revision "2017-07-03" {
       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 {
       description
        "The 'tls-initiate' feature indicates that the RESTCONF client
         supports initiating TLS connections to RESTCONF servers.";
       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 {
       description
        "The 'tls-listen' feature indicates that the RESTCONF client
         supports opening a port to listen for incoming RESTCONF
         server call-home TLS connections.";
       reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
```

```
  }

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

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

    container initiate {
      if-feature initiate;
      description
        "Configures client initiating underlying TCP connections.";
      list restconf-server {
        key name;
        description
          "List of RESTCONF servers the RESTCONF client is to initiate
           connections to.";
        leaf name {
          type string;
          description
            "An arbitrary name for the RESTCONF server.";
        }
        choice transport {
          mandatory true;
          description
            "Selects between available transports.";

          case tls {
            if-feature tls-initiate;
            container tls {
              description
                "Specifies TLS-specific transport configuration.";
              uses endpoints-container {
                refine endpoints/endpoint/port {
                  default 443;
                }
              }
              uses ts:tls-client-grouping {
                refine "client-auth/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 {
            description
              "Selects between available connection types.";
            case persistent-connection {
              container persistent {
                presence true;
                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.";
                leaf idle-timeout {
                  type uint32;
                  units "seconds";
                  default 86400;  // one day;
                  description
                    "Specifies the maximum number of seconds that a
                     a RESTCONF session may remain idle. A RESTCONF
                     session will be dropped if it is idle for an
                     interval longer than this number of seconds.
                     If set to zero, then the client will never drop
                     a session because it is idle.  Sessions that
                     have a notification subscription active are
                     never dropped.";
                }
                container keep-alives {
                  description
                    "Configures the keep-alive policy, to proactively
                     test the aliveness of the SSH/TLS server.  An
                     unresponsive SSH/TLS server will be dropped after
                     approximately max-attempts * max-wait seconds.";
                  reference
                    "RFC 8071: NETCONF Call Home and RESTCONF Call
                     Home, Section 3.1, item S6";
                  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 SSH/TLS
                      server, a SSH/TLS-level message will be sent
                      to test the aliveness of the SSH/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 SSH/TLS server before assuming the SSH/TLS
                      server is no longer alive.";
                  }
                }
              }
            }
            case periodic-connection {
              container periodic {
                presence true;
                description
                 "Periodically connect to the RESTCONF server, so that
                  the RESTCONF server may deliver messages pending for
                  the RESTCONF client.  The RESTCONF server must close
                  the connection when it is ready to release it. Once
                  the connection has been closed, the RESTCONF client
                  will restart its timer until the next connection.";
                leaf idle-timeout {
                  type uint16;
                  units "seconds";
                  default 300; // five minutes
                  description
                    "Specifies the maximum number of seconds that a
                     a RESTCONF session may remain idle. A RESTCONF
                     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.";
                }
                leaf reconnect-timeout {
```

```
                  type uint16 {
                    range "1..max";
                  }
                  units minutes;
                  default 60;
                  description
                   "Sets the maximum amount of unconnected time the
                    RESTCONF client will wait before re-establishing
                    a connection to the RESTCONF server.  The RESTCONF
                    client may initiate a connection before this
                    time if desired (e.g., to set configuration).";
                }
              }
            }
          }
        }
        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.";
              }
            }
            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 restconf-server
    } // end initiate

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

      leaf max-sessions {
        type uint16;
        default 0;
        description
          "Specifies the maximum number of concurrent sessions
           that can be active at one time.  The value 0 indicates
           that no artificial session limit should be used.";
      }

      leaf idle-timeout {
        type uint16;
        units "seconds";
        default 3600; // one hour
        description
          "Specifies the maximum number of seconds that a RESTCONF
           session may remain idle. A RESTCONF 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;
        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.";
          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 for call-home connections.";
              }
              leaf port {
                type inet:port-number;
                default 4336;
                description
                 "The port number to listen for call-home connections.";
              }
              uses ts:tls-client-grouping {
                refine "client-auth/auth-type" {
                  mandatory true;
                  description
                    "RESTCONF clients MUST pass some authentication
                     credentials.";
                }
              }
            }
          }
        } // end transport
      } // end endpoint
    } // end listen

  } // end restconf-client


  grouping endpoints-container {
    description
      "This grouping is used to configure a set of RESTCONF servers
       a RESTCONF client may initiate connections to.";
    container endpoints {
      description
        "Container for the list of endpoints.";
      list endpoint {
```

```
        key name;
        unique "address port";
        min-elements 1;
        ordered-by user;
        description
          "A non-empty user-ordered list of endpoints for this RESTCONF
           client to try to connect to.  Defining more than one enables
           high-availability.";
        leaf name {
          type string;
          description
            "An arbitrary name for this endpoint.";
        }
        leaf address {
          type inet:host;
          mandatory true;
          description
           "The IP address or hostname of the endpoint.  If a
            hostname is configured and the DNS resolution results
            in more than one IP address, the RESTCONF client
            will process the IP addresses as if they had been
            explicitly configured in place of the hostname.";
        }
        leaf port {
          type inet:port-number;
          description
           "The IP port for this endpoint. The RESTCONF client will
            use the IANA-assigned well-known port (set via a refine
            statement when uses) if no value is specified.";
        }
      }
    }
  }
}

<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.

This model supports the TLS transport protocol using the TLS server groupings defined in [I-D.ietf-netconf-tls-client-server].

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

Just the container is displayed below, but there is also a grouping that the container is using.

Note: all lines are folded at column 71 with no '\' character.

```
module: ietf-restconf-server
    +--rw restconf-server
       +--rw listen {listen}?
       |  +--rw max-sessions?   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 certificates
       |              |  +--rw certificate* [name]
       |              |     +--rw name    leafref
       |              +--rw client-auth
       |              |  +--rw trusted-ca-certs?       leafref
       |              |  +--rw trusted-client-certs?   leafref
       |              |  +--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 (transport)
             |  +--:(tls) {tls-call-home}?
             |     +--rw tls
             |        +--rw endpoints
             |        |  +--rw endpoint* [name]
             |        |     +--rw name       string
```

```
             |        |     +--rw address    inet:host
             |        |     +--rw port?      inet:port-number
             |        +--rw certificates
             |        |  +--rw certificate* [name]
             |        |     +--rw name    leafref
             |        +--rw client-auth
             |        |  +--rw trusted-ca-certs?       leafref
             |        |  +--rw trusted-client-certs?   leafref
             |        |  +--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 reconnect-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 2.2 of [I-D.ietf-netconf-keystore].

```
<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">

  <!-- listening for TLS (HTTPS) connections -->
```

```
  <listen>
    <endpoint>
      <name>netconf/tls</name>
      <tls>
        <address>11.22.33.44</address>
        <certificates>
          <certificate>
            <name>tls-ec-cert</name>
          </certificate>
        </certificates>
        <client-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-client-certs>explicitly-trusted-client-certs</trusted-
client-certs>
          <cert-maps>
            <cert-to-name>
              <id>1</id>
              <fingerprint>11:0A:05:11:00</fingerprint>
              <map-type>x509c2n:san-any</map-type>
            </cert-to-name>
            <cert-to-name>
              <id>2</id>
              <fingerprint>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>

  <!-- calling home to a RESTCONF client -->
  <call-home>
    <restconf-client>
      <name>config-manager</name>
      <tls>
        <endpoints>
          <endpoint>
            <name>east-data-center</name>
            <address>22.33.44.55</address>
          </endpoint>
          <endpoint>
            <name>west-data-center</name>
            <address>33.44.55.66</address>
          </endpoint>
        </endpoints>
        <certificates>
```

```
          <certificate>
```

```
            <name>tls-ec-cert</name>
          </certificate>
        </certificates>
        <client-auth>
          <trusted-ca-certs>deployment-specific-ca-certs</trusted-ca-certs>
          <trusted-client-certs>explicitly-trusted-client-certs</trusted-
client-certs>
          <cert-maps>
            <cert-to-name>
              <id>1</id>
              <fingerprint>11:0A:05:11:00</fingerprint>
              <map-type>x509c2n:san-any</map-type>
            </cert-to-name>
            <cert-to-name>
              <id>2</id>
              <fingerprint>B3:4F:A1:8C:54</fingerprint>
              <map-type>x509c2n:specified</map-type>
              <name>scooby-doo</name>
            </cert-to-name>
          </cert-maps>
        </client-auth>
      </tls>
      <connection-type>
        <periodic>
          <idle-timeout>300</idle-timeout>
          <reconnect-timeout>60</reconnect-timeout>
        </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 Model

This YANG module imports YANG types from [RFC6991] and [RFC7407].

```
<CODE BEGINS> file "ietf-restconf-server@2017-07-03.yang"

module ietf-restconf-server {
  yang-version 1.1;

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

```
//import ietf-netconf-acm {
//  prefix nacm;
//  reference
//    "RFC 6536: Network Configuration Protocol (NETCONF)
//     Access Control Model";
//}

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 2017-06-13; // stable grouping definitions
  reference
    "RFC ZZZZ: TLS Client and Server Models";
}

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

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

  WG Chair: Mehmet Ersue
            <mailto:mehmet.ersue@nsn.com>

  WG Chair: Mahesh Jethanandani
            <mailto:mjethanandani@gmail.com>

  Editor:   Kent Watsen
            <mailto:kwatsen@juniper.net>";


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

  Copyright (c) 2014 IETF Trust and the persons identified as
```

```
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD
    License set forth in Section 4.c of the IETF Trust's
    Legal Provisions Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";

  revision "2017-07-03" {
    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 {
    description
     "The 'tls-listen' feature indicates that the RESTCONF server
      supports opening a port to listen for incoming RESTCONF
      client connections.";
    reference
     "RFC XXXX: RESTCONF Protocol";
  }

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

  feature tls-call-home {
```

```
    description
     "The 'tls-call-home' feature indicates that the RESTCONF server
      supports initiating connections to RESTCONF clients.";
    reference
     "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
  }

  feature client-cert-auth {
    description
     "The client-cert-auth feature indicates that the RESTCONF
      server supports the ClientCertificate authentication scheme.";
    reference
     "RFC ZZZZ: Client Authentication over New TLS Connection";
  }


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

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

    container listen {
      if-feature listen;
      description
        "Configures listen behavior";
      leaf max-sessions {
        type uint16;
        default 0;   // should this be 'max'?
        description
          "Specifies the maximum number of concurrent sessions
           that can be active at one time.  The value 0 indicates
           that no artificial session limit should be used.";
      }
      list endpoint {
        key name;
        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.";
          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 of the interface to listen on.  The
                  TLS server will listen on all interfaces if no value
                  is specified.  Please note that some addresses have
                  special meanings (e.g., '0.0.0.0' and '::').";
              }
              leaf port {
                type inet:port-number;
                default 443;
                description
                 "The local port number on this interface the TLS server
                  listens on.";
              }
              uses ts:tls-server-grouping {
                refine "client-auth" {
                  must 'trusted-ca-certs or trusted-client-certs';
                  description
                    "RESTCONF servers MUST be able to validate clients.";
                }
                augment "client-auth" {
                  description
                    "Augments in the cert-to-name structure.";
                  uses cert-maps-grouping;
                }
              }
            } // end tls container
          } // end tls case
        } // end transport
      } // end endpoint
    } // end listen

    container call-home {
      if-feature call-home;
      description
        "Configures call-home behavior";
      list restconf-client {
```

```
        key name;
        description
          "List of RESTCONF clients the RESTCONF server is to
           initiate call-home connections to.";
        leaf name {
          type string;
          description
            "An arbitrary name for the remote RESTCONF client.";
        }
        choice transport {
          mandatory true;
          description
            "Selects between TLS and any transports augmented in.";
          case tls {
            if-feature tls-call-home;
            container tls {
              description
                "Specifies TLS-specific call-home transport
                 configuration.";
              uses endpoints-container {
                refine endpoints/endpoint/port {
                  default 4336;
                }
              }
              uses ts:tls-server-grouping {
                refine "client-auth" {
                  must 'trusted-ca-certs or trusted-client-certs';
                  description
                    "RESTCONF servers MUST be able to validate clients.";
                }
                augment "client-auth" {
                  description
                    "Augments in the cert-to-name structure.";
                  uses cert-maps-grouping;
                }
              }
            }
          }
        }
        container connection-type {
          description
           "Indicates the RESTCONF client's preference for how the
            RESTCONF server's connection is maintained.";
          choice connection-type {
            description
              "Selects between available connection types.";
            case persistent-connection {
              container persistent {
```

```
                presence true;
                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 3.1, item S6";
                  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 true;
```

```
                description
                 "Periodically connect to the RESTCONF client, so that
                  the RESTCONF client may deliver messages pending for
                  the RESTCONF server.  The client must close the
                  connection when it's ready to release it.  Once the
                  connection has been closed, the server will restart
                  its timer until the next connection.";
                leaf reconnect-timeout {
                  type uint16 {
                    range "1..max";
                  }
                  units minutes;
                  default 60;
                  description
                   "The maximum amount of unconnected time the
                    RESTCONF server will wait before re-establishing
                    a connection to the RESTCONF client.  The
                    RESTCONF server may initiate a connection to
                    the RESTCONF client before this time if desired
                    (e.g., to deliver a notification).";
                }
              }
            }
          }
        }
        container reconnect-strategy {
          description
           "The reconnection strategy directs how a RESTCONF server
            reconnects to a RESTCONF client after 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.";
              }
            }
            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).";
          }
        }
      }
    }
  }


  grouping cert-maps-grouping {
    description
      "A grouping that defines a container around the
       cert-to-name structure defined in RFC 7407.";
    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 XXXX: The RESTCONF Protocol";
    }
  }


  grouping endpoints-container {
    description
      "This grouping is used by tls container for call-home
       configurations.";
```

```
    container endpoints {
      description
        "Container for the list of endpoints.";
      list endpoint {
        key name;
        unique "address port";
        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.";
        }
        leaf address {
          type inet:host;
          mandatory true;
          description
           "The IP address or hostname of the endpoint.  If a
            hostname is configured and the DNS resolution results
            in more than one IP address, the RESTCONF server
            will process the IP addresses as if they had been
            explicitly configured in place of the hostname.";
        }
        leaf port {
          type inet:port-number;
          description
           "The IP port for this endpoint. The RESTCONF server will
            use the IANA-assigned well-known port if no value is
            specified.";
        }
      }
    }
  }


}


<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) [RFC6536] 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:

NONE

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 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 [RFC7950]. Following the format in [RFC7950], 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. 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, Phil Shafer, Sean Turner, and Bert Wijnen.

Juergen Schoenwaelder and was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

7. References

7.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "Keystore Model", draft-ietf-netconf-keystore-02 (work in progress), June 2017.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "TLS Client and Server Models", draft-ietf-netconf-tls-client-server-03 (work in progress), June 2017.

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

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7950>.

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

7.2. Informative References

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[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, <http://www.rfc-editor.org/info/rfc6241>.

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

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

Appendix A. Change Log

A.1. server-model-09 to 00

- This draft was split out from draft-ietf-netconf-server-model-09.

- Added in new features 'listen' and 'call-home' so future transports can be augmented in.

A.2. 00 to 01

- Renamed "keychain" to "keystore".

A.3. 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.4. 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.5. 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.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Juergen Schoenwaelder
Jacobs University Bremen

EMail: j.schoenwaelder@jacobs-university.de