NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

K. Watsen
Juniper Networks
J. Schoenwaelder
Jacobs University Bremen

July 8, 2016

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-00

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:

- draft-ietf-netconf-system-keychain
- draft-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
- "YYYY" --> the assigned RFC value for draft-ietf-netconf-restconf
- "ZZZZ" --> the assigned RFC value for draft-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:

- "2016-07-08" --> the publication date of this draft

The following two Appendix sections are to be removed prior to publication:

- Appendix A. Change Log
- Appendix B. Open Issues

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines two YANG [RFC6020] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server [draft-ietf-netconf-restconf]. Both modules support the TLS [RFC5246] transport protocol with both standard RESTCONF and RESTCONF Call Home connections [draft-ietf-netconf-call-home].

1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 both TLS transport protocols using the TLS client groupings defined in [draft-ietf-netconf-tls-client-server].

All private keys and trusted certificates are held in the keychain model defined in [draft-ietf-netconf-system-keychain].

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

2.1. Tree Diagram

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

```
module: ietf-restconf-client
   +--rw restconf-client
      +--rw initiate {tls-initiate}?
      +--rw listen {tls-listen}?
```

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 [draft-ietf-netconf-system-keychain].

FIXME

2.3. YANG Model

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

<CODE BEGINS> file "ietf-restconf-client@2016-07-08.yang"

```
   // Editor's Note:
   // This module is incomplete at this time.  Below is
   // just a skeleton so there's something in the draft.
   // Please ignore this module for now!

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

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

     import ietf-tls-client {
       prefix ts;
       revision-date 2016-07-08; // 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 "2016-07-08" {
       description
        "Initial version";
       reference
        "RFC XXXX: RESTCONF Client and Server Models";
     }

     // Features

     feature tls-initiate {
       description
        "The tls-initiate feature indicates that the RESTCONF client
         supports initiating TLS connections to RESTCONF servers.";
       reference
        "RFC YYYY: RESTCONF Protocol";
     }

     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 AAAA: NETCONF Call Home and RESTCONF Call Home";
```

```
    }

    container restconf-client {
      description
        "Top-level container for RESTCONF client configuration.";
      container initiate {
        if-feature tls-initiate;
        description
          "Configures client intiating underlying TCP connections.";
      }
      container listen {
        if-feature tls-listen;
        description
          "Configures client accepting call-home TCP connections.";
      }
    }

  }


  <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 using the TLS server groupings defined in [draft-ietf-netconf-tls-client-server].

All private keys and trusted certificates are held in the keychain model defined in [draft-ietf-netconf-system-keychain].

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

3.1. Tree Diagram

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    -> /kc:keychain/private-keys/p
rivate-key/certificate-chains/certificate-chain/name
      |              +--rw client-auth
      |                 +--rw trusted-ca-certs?       -> /kc:keychain/t
rusted-certificates/name
      |                 +--rw trusted-client-certs?   -> /kc:keychain/t
rusted-certificates/name
      |                 +--rw cert-maps
      |                    +--rw cert-to-name* [id]
      |                       +--rw id             uint32
      |                       +--rw fingerprint    x509c2n:tls-fingerpr
int
      |                       +--rw map-type       identityref
      |                       +--rw name           string
      +--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    -> /kc:keychain/private-keys/p
rivate-key/certificate-chains/certificate-chain/name
            |        +--rw client-auth
            |           +--rw trusted-ca-certs?       -> /kc:keychain/t
rusted-certificates/name
            |           +--rw trusted-client-certs?   -> /kc:keychain/t
rusted-certificates/name
            |           +--rw cert-maps
            |              +--rw cert-to-name* [id]
            |                 +--rw id             uint32
            |                 +--rw fingerprint    x509c2n:tls-fingerpr
int
            |                 +--rw map-type       identityref
            |                 +--rw name           string
            +--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 [draft-ietf-netconf-system-keychain].

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

  <!-- listening for TLS (HTTPS) connections -->
  <listen>
    <endpoint>
      <name>netconf/tls</name>
      <tls>
        <address>11.22.33.44</address>
        <certificates>
          <certificate>ex-key-sect571r1-cert</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>ex-key-sect571r1-cert</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@2016-07-08.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 2016-07-08; // 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 "2016-07-08" {
      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 YYYY: 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 YYYY: 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 {
      description
        "Top-level container 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 on.";
          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.";
                uses ts:listening-tls-server-grouping {
                  refine port {
                    default 443;
                  }
                  augment "client-auth" {
                    description
                      "Augments in the cert-to-name structure.";
                    uses cert-maps-grouping;
                  }
                }
              }
            }
```

```
          }
        }
      }

      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:non-listening-tls-server-grouping {
                  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 YYYY: 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;
          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

5. IANA Considerations

5.1. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC2119]. 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. 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, 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

[draft-ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ieft-netconf-restconf-13 (work in progress), 2016, <https://datatracker.ietf.org/html/draft-ieft-netconf-restconf>.

[draft-ietf-netconf-system-keychain]
Watsen, K., "System Keychain Model", draft-ieft-netconf-system-keychain-00 (work in progress), 2016, <https://datatracker.ietf.org/html/draft-ieft-netconf-system-keychain>.

[draft-ietf-netconf-tls-client-server]
Watsen, K., "TLS Client and Server Models", draft-ieft-netconf-tls-client-server-00 (work in progress), 2016, <https://datatracker.ietf.org/html/draft-ieft-netconf-tls-client-server>.

[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>.

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

[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>.

7.2. Informative References

[draft-ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ieft-netconf-call-home-17 (work in progress), 2015, <https://datatracker.ietf.org/html/draft-ieft-netconf-call-home-17>.

[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>.

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.

Appendix B. Open Issues

Please see: https://github.com/netconf-wg/restconf-client-server/issues.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Juergen Schoenwaelder
Jacobs University Bremen

EMail: j.schoenwaelder@jacobs-university.de