

A YANG Data Model for RESTCONF Clients and Servers
draft-ietf-netconf-restconf-client-server-43

Abstract

This document presents two YANG modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server. These modules support both standard and call home RESTCONF connections. For initiating connections, both modules configure HTTPS. For listening for connections, both modules configure HTTPS and HTTP. Whilst RESTCONF supports only HTTPS, HTTP may be configured for when a TLS-terminator is deployed in front of the listener.

Editorial Note (To be removed by RFC Editor)

This draft contains 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.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements (note: not all may be present):

- * GGGG --> the assigned RFC value for draft-ietf-netconf-http-client-server
- * HHHH --> the assigned RFC value for draft-ietf-netconf-netconf-client-server
- * IIII --> the assigned RFC value for this draft
- * JJJJ --> the assigned RFC value for draft-ietf-netconf-udp-client-server

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

- * 2025-04-24 --> the publication date of this draft

The "Relation to other RFCs" section Section 1.1 contains the text "one or more YANG modules" and, later, "modules". This text is sourced from a file in a context where it is unknown how many modules a draft defines. The text is not wrong as is, but it may be improved by stating more directly how many modules are defined.

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\ ' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\ ' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\ ' folded examples to use the '\\ ' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

Due to a bug in the pyang tool used to create tree diagrams, some "key" nodes appear as optional (i.e., have a '?' postfix). Ideally the '?' character is removed in the tree diagrams for "key" nodes. Recipe: search for lists using the string "* [" , then note the nodes appearing in the square brackets (e.g., "* [name]"), then look for matching child nodes and remove the '?' characters (e.g., "name?" becomes "name").

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 26 October 2025.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
1.1. Relation to other RFCs	5
1.2. Specification Language	6
1.3. Adherence to the NMDA	6
2. The "ietf-restconf-client" Module	6
2.1. Data Model Overview	7
2.2. Example Usage	10
2.3. YANG Module	14
3. The "ietf-restconf-server" Module	24
3.1. Data Model Overview	24
3.2. Example Usage	29
3.3. YANG Module	35
4. Security Considerations	49
4.1. Considerations for the "ietf-restconf-client" YANG Module	49
4.2. Considerations for the "ietf-restconf-server" YANG Module	50
5. IANA Considerations	51
5.1. The "IETF XML" Registry	51
5.2. The "YANG Module Names" Registry	51
6. References	51
6.1. Normative References	51
6.2. Informative References	53
Appendix A. Change Log	54
A.1. 00 to 01	54
A.2. 01 to 02	54
A.3. 02 to 03	54
A.4. 03 to 04	55
A.5. 04 to 05	55
A.6. 05 to 06	55
A.7. 06 to 07	55
A.8. 07 to 08	56

A.9.	08 to 09	56
A.10.	09 to 10	56
A.11.	10 to 11	56
A.12.	11 to 12	56
A.13.	12 to 13	57
A.14.	13 to 14	57
A.15.	14 to 15	57
A.16.	15 to 16	58
A.17.	16 to 17	58
A.18.	17 to 18	58
A.19.	18 to 19	58
A.20.	19 to 20	58
A.21.	20 to 21	59
A.22.	21 to 22	59
A.23.	22 to 23	59
A.24.	23 to 24	59
A.25.	24 to 25	59
A.26.	25 to 26	59
A.27.	26 to 27	59
A.28.	27 to 28	60
A.29.	28 to 29	60
A.30.	29 to 30	60
A.31.	30 to 31	60
A.32.	31 to 32	60
A.33.	32 to 34	60
A.34.	34 to 36	61
A.35.	38 to 39	61
A.36.	39 to 40	61
A.37.	40 to 41	61
A.38.	41 to 42	61
A.39.	42 to 43	62
Acknowledgements		62
Author's Address		62

1. Introduction

This document presents two YANG [RFC7950] modules, one module to configure a RESTCONF client and the other module to configure a RESTCONF server. These modules support both standard [RFC8040] and Call Home [RFC8071] RESTCONF connections.

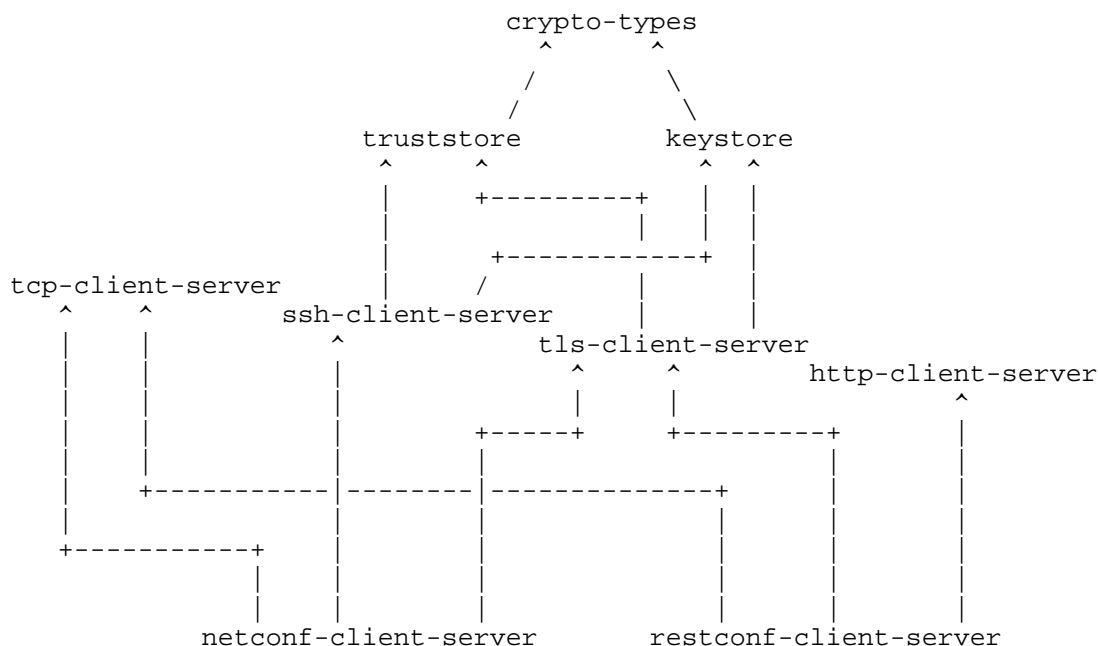
For initiating connections, both modules configure HTTPS, whether by using TLS [RFC8446] or QUIC [RFC9000]. For when listening for connections, both modules configure HTTPS and HTTP, where HTTP is used when a TLS-terminator is deployed in front of the listener.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, support the configuration of both the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

Primary dependency relationships between the YANG groupings defined in the various RFCs is presented in the below diagram. In some cases, a draft may define secondary groupings that introduce dependencies not illustrated in the diagram. The labels in the diagram are a shorthand name for the defining RFC. The citation reference for shorthand name is provided below the diagram.

Please note that the arrows in the diagram point from referencer to referenced. For example, the "crypto-types" RFC does not have any dependencies, whilst the "keystore" RFC depends on the "crypto-types" RFC.



Label in Diagram	Originating RFC
crypto-types	[RFC9640]
truststore	[RFC9641]
keystore	[RFC9642]
tcp-client-server	[RFC9643]
ssh-client-server	[RFC9644]
tls-client-server	[RFC9645]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label in Diagram to RFC Mapping

1.2. Specification Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [RFC9641] and [RFC9642], trust anchors and keys installed during manufacturing are expected to appear in <operational> (Section 5.3 of [RFC8342]), and <system> [I-D.ietf-netmod-system-config], if implemented.

2. The "ietf-restconf-client" Module

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

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

2.1. Data Model Overview

This section provides an overview of the "ietf-restconf-client" module in terms of its features and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-restconf-client" module:

Features:

```
+-- https-initiate
+-- http-listen
+-- https-listen
+-- central-restconf-client-supported
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

2.1.2. Groupings

The "ietf-restconf-client" module defines the following "grouping" statements:

```
* restconf-client-initiate-stack-grouping
* restconf-client-listen-stack-grouping
* restconf-client-app-grouping
```

Each of these groupings are presented in the following subsections.

2.1.2.1. The "restconf-client-initiate-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-initiate-stack-grouping" grouping:

```
grouping restconf-client-initiate-stack-grouping:
  +-- http-client-parameters
  |   +---u httpc:http-client-grouping
  +-- restconf-client-parameters
      +---u rcc:restconf-client-grouping
```

Comments:

- * The "restconf-client-initiate-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that initiate connections to RESTCONF servers, as opposed to receiving call-home [RFC8071] connections.
- * The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.
- * For the referenced grouping statement(s):
 - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [RFC9643].
 - The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [RFC9645].
 - The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].

2.1.2.2. The "restconf-client-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-listen-stack-grouping" grouping:

```
grouping restconf-client-listen-stack-grouping:
  +-- (transport)
    +--:(tcp) {tcp-listen}?
      +-- tcp
        +-- tcp-server-parameters
          | +---u tcps:tcp-server-grouping
        +-- http-client-parameters
          | +---u httpc:http-client-grouping
        +-- restconf-client-parameters
          +---u rcc:restconf-client-grouping
```

Comments:

- * The "restconf-client-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF clients that receive call-home [RFC8071] connections from RESTCONF servers.
- * The "transport" choice node enables either the HTTP or HTTPS transports to be configured, with each option enabled by a "feature" statement. Whilst RESTCONF [RFC8040] requires HTTPS, an HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-client.
- * For the referenced grouping statement(s):

- The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [RFC9643].
- The "tls-client-grouping" grouping is discussed in Section 3.1.2.1 of [RFC9645].
- The "http-client-grouping" grouping is discussed in Section 2.1.2.2 of [I-D.ietf-netconf-http-client-server].

2.1.2.3. The "restconf-client-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-client-app-grouping" grouping:

```

grouping restconf-client-app-grouping:
  +-- initiate! {initiate}?
  |   +-- restconf-server* [name]
  |   |   +-- name? string
  |   |   +-- endpoints
  |   |   |   +-- endpoint* [name]
  |   |   |   |   +-- name? string
  |   |   |   |   +---u restconf-client-initiate-stack-grouping
  |   |   +-- connection-type
  |   |   |   +-- (connection-type)
  |   |   |   |   +--:(persistent-connection)
  |   |   |   |   |   +-- persistent!
  |   |   |   |   +--:(periodic-connection)
  |   |   |   |   |   +-- periodic!
  |   |   |   |   |   +-- period? uint16
  |   |   |   |   |   +-- anchor-time? yang:date-and-time
  |   |   |   |   |   +-- idle-timeout? uint16
  |   |   +-- reconnect-strategy
  |   |   |   +-- start-with? enumeration
  |   |   |   +-- max-wait? uint16
  |   |   |   +-- max-attempts? uint8
  |   +-- listen! {listen}?
  |   |   +-- idle-timeout? uint16
  |   |   +-- endpoints
  |   |   |   +-- endpoint* [name]
  |   |   |   |   +-- name? string
  |   |   |   |   +---u restconf-client-listen-stack-grouping

```

Comments:

- * The "restconf-client-app-grouping" defines the configuration for a RESTCONF client that supports both initiating connections to RESTCONF servers as well as receiving call-home connections from RESTCONF servers.

- * Both the "initiate" and "listen" subtrees are predicated by "feature" statements.
- * For the referenced grouping statement(s):
 - The "restconf-client-initiate-stack-grouping" grouping is discussed in Section 2.1.2.1 in this document.
 - The "restconf-client-listen-stack-grouping" grouping is discussed in Section 2.1.2.2 in this document.

2.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-restconf-client" module:

```
module: ietf-restconf-client
  +--rw restconf-client {central-restconf-client-supported}?
    +---u restconf-client-app-grouping
```

Comments:

- * Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- * The top-level node "restconf-client" is additionally constrained by the feature "central-restconf-client-supported".
- * The "restconf-client-app-grouping" grouping is discussed in Section 2.1.2.3 in this document.
- * The reason for why "restconf-client-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of restconf-client-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

2.2. Example Usage

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

This example is consistent with the examples presented in Section 2.2.1 of [RFC9641] and Section 2.2.1 of [RFC9642].

===== NOTE: '\ ' line wrapping per RFC 8792 =====

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

```
<!-- RESTCONF servers to initiate connections to -->
```

```
<initiate>
```

```
<restconf-server>
```

```
<name>corp-fw1</name>
```

```
<endpoints>
```

```
<endpoint>
```

```
<name>corp-fw1.example.com</name>
```

```
<http-client-parameters>
```

```
<uri>
```

```
<scheme>https</scheme>
```

```
<authority>
```

```
<host>corp-fw1.example.com</host>
```

```
</authority>
```

```
</uri>
```

```
<tls-client-parameters>
```

```
<client-identity>
```

```
<certificate>
```

```
<central-keystore-reference>
```

```
<asymmetric-key>rsa-asymmetric-key</asymmetric-k\
```

```
ey>
```

```
<certificate>ex-rsa-cert</certificate>
```

```
</central-keystore-reference>
```

```
</certificate>
```

```
</client-identity>
```

```
<server-authentication>
```

```
<ca-certs>
```

```
<central-truststore-reference>trusted-server-ca-ce\
```

```
rts</central-truststore-reference>
```

```
</ca-certs>
```

```
<ee-certs>
```

```
<central-truststore-reference>trusted-server-ee-ce\
```

```
rts</central-truststore-reference>
```

```
</ee-certs>
```

```
</server-authentication>
```

```
<keepalives>
```

```
<test-peer-aliveness>
```

```
<max-wait>30</max-wait>
```

```
<max-attempts>3</max-attempts>
```

```
</test-peer-aliveness>
```

```
</keepalives>
```

```
</tls-client-parameters>
```

```
</http-client-parameters>
```

```
</endpoint>
```

```

    <endpoint>
      <name>corp-fw2.example.com</name>
      <http-client-parameters>
        <uri>
          <scheme>https</scheme>
          <authority>
            <host>corp-fw2.example.com</host>
          </authority>
        </uri>
        <tls-client-parameters>
          <client-identity>
            <certificate>
              <central-keystore-reference>
                <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                <certificate>ex-rsa-cert</certificate>
              </central-keystore-reference>
            </certificate>
          </client-identity>
          <server-authentication>
            <ca-certs>
              <central-truststore-reference>trusted-server-ca-ce\
rts</central-truststore-reference>
            </ca-certs>
            <ee-certs>
              <central-truststore-reference>trusted-server-ee-ce\
rts</central-truststore-reference>
            </ee-certs>
          </server-authentication>
        </keepalives>
        <test-peer-aliveness>
          <max-wait>30</max-wait>
          <max-attempts>3</max-attempts>
        </test-peer-aliveness>
      </keepalives>
    </tls-client-parameters>
  </http-client-parameters>
</endpoint>
</endpoints>
<connection-type>
  <persistent/>
</connection-type>
</restconf-server>
</initiate>

<!-- endpoints to listen for RESTCONF Call Home connections on -->
<listen>
  <endpoints>

```

```

    <endpoint>
      <name>Intranet-facing listener</name>
      <tcp>
        <tcp-server-parameters>
          <local-bind>
            <local-address>192.0.2.2</local-address>
          </local-bind>
        </tcp-server-parameters>
        <http-client-parameters>
          <uri>
            <scheme>https</scheme>
            <authority>
              <host>corp-fw1.example.com</host>
            </authority>
          </uri>
          <tls-client-parameters>
            <client-identity>
              <certificate>
                <central-keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                  <certificate>ex-rsa-cert</certificate>
                </central-keystore-reference>
              </certificate>
            </client-identity>
            <server-authentication>
              <ca-certs>
                <central-truststore-reference>trusted-server-ca-ce\
rts</central-truststore-reference>
              </ca-certs>
              <ee-certs>
                <central-truststore-reference>trusted-server-ee-ce\
rts</central-truststore-reference>
              </ee-certs>
            </server-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-client-parameters>
        </http-client-parameters>
      </tcp>
    </endpoint>
  </endpoints>
</listen>
</restconf-client>

```

2.3. YANG Module

This YANG module has normative references to [RFC6991], [RFC8040], and [RFC8071], [RFC9643], [RFC9645], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-client@2025-04-24.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-tcp-server {
    prefix tcps;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-http-client {
    prefix httpc;
    reference
      "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
  }

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

  contact
    "WG Web:  https://datatracker.ietf.org/wg/netconf
    WG List:  NETCONF WG list <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

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

    Copyright (c) 2025 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 Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC IIII (<https://www.rfc-editor.org/info/rfcIIII>); see the RFC itself for full legal notices.

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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.";

```
revision 2025-04-24 {
  description
    "Initial version";
  reference
    "RFC IIII: A YANG Data Model for RESTCONF Clients and Server";
}

// Features

feature initiate {
  description
    "The 'initiate' feature indicates that the RESTCONF
    client supports initiating connections to RESTCONF
    servers.";
  reference
    "RFC 8040: RESTCONF Protocol";
}

feature listen {
  description
    "The 'listen' feature indicates that the RESTCONF client
    supports listening for incoming RESTCONF server call-home
    connections.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

feature tcp-listen {
  if-feature listen;
  description
    "The 'tcp-listen' feature indicates that the RESTCONF client
    supports listening for incoming RESTCONF server call-home
```

```
        connections over TCP (e.g., HTTP/1 or HTTP/2).";
    reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

feature central-restconf-client-supported {
    description
        "The 'central-restconf-client-supported' feature indicates
        that the server that implements this module supports
        the top-level 'restconf-client' node.

        This feature is needed as some servers may want to use
        features defined in this module, which requires this
        module to be implemented, without having to support
        the top-level 'restconf-client' node.";
}

feature foo {
    description
        "This is an internal feature used only to create a
        logical fallacy to remove unwanted nodes from the
        'ietf-http-client' grouping.";
}

// Groupings

grouping restconf-client-grouping {
    description
        "A grouping for configuring a RESTCONF client without any
        consideration for how underlying transport sessions are
        established.

        As no RESTCONF-specific client configuration needs to be
        set, this node is empty, but retained for clarity and for
        consistency with other 'client-server' models.";
}

grouping restconf-client-initiate-stack-grouping {
    description
        "A grouping for configuring a RESTCONF client
        'initiate' protocol stack for a single outbound connection.";
    container http-client-parameters {
        description
            "HTTP-level client parameters.";
        uses httpc:http-client-grouping {
            refine "uri/scheme" {
                must '. = "https"';
            }
        }
    }
}
```



```
        description
            "RFC 8040 requires the 'https' scheme.";
    }
    refine 'uri/path' {
        if-feature "foo and not foo"; // logically impossible
        description
            "The 'path' component of the URI does not make
            sense to configure for a RESTCONF client.";
    }
    refine 'uri/query' {
        if-feature "foo and not foo"; // logically impossible
        description
            "The 'query' component of the URI does not make
            sense to configure for a RESTCONF client.";
    }
    refine 'uri/fragment' {
        if-feature "foo and not foo"; // logically impossible
        description
            "The 'fragment' component of the URI does not make
            sense to configure for a RESTCONF client.";
    }
    }
}
container restconf-client-parameters {
    description
        "RESTCONF-level client parameters to initiate
        a RESTCONF over HTTP connection.";
    uses rcc:restconf-client-grouping;
}

} // restconf-client-initiate-stack-grouping

grouping restconf-client-listen-stack-grouping {
    description
        "A grouping for configuring a RESTCONF client to listen
        to a port for call home connections, as described in
        RFC 8071.";
    reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    choice transport {
        mandatory true;
        description
            "Selects between available transports.";
        case tcp {
            if-feature "tcp-listen";
            container tcp {
                description
```

```
"Configures the client to open a TCP port to listen for
TCP-client connections, i.e., RESTCONF-server call home
connections.";
container tcp-server-parameters {
  description
    "TCP-level server parameters.";
  uses tcps:tcp-server-grouping {
    refine "local-bind/local-port" {
      default "4336";
      description
        "The RESTCONF client will listen on the IANA-
        assigned well-known port for 'restconf-ch-tls'
        (4336) if no value is specified.";
    }
  }
}
container http-client-parameters {
  description
    "HTTP-level client parameters.";
  uses httpc:http-client-grouping {
    refine "uri/scheme" {
      must '. = "https"';
      description
        "RFC 8040 requires the 'https' scheme.";
    }
    refine 'uri/authority/port' {
      if-feature "foo and not foo"; // logically impossible
      description
        "The authority's 'port' component of the URI does
        not make sense to configure for a RESTCONF client
        listening for a call home connections.";
    }
    refine 'uri/path' {
      if-feature "foo and not foo"; // logically impossible
      description
        "The 'path' component of the URI does not make
        sense to configure for a RESTCONF client.";
    }
    refine 'uri/query' {
      if-feature "foo and not foo"; // logically impossible
      description
        "The 'query' component of the URI does not make
        sense to configure for a RESTCONF client.";
    }
    refine 'uri/fragment' {
      if-feature "foo and not foo"; // logically impossible
      description
        "The 'fragment' component of the URI does not make
```

```
        sense to configure for a RESTCONF client.";
    }
}
}
container restconf-client-parameters {
    description
        "RESTCONF-level client parameters.";
    uses rcc:restconf-client-grouping;
}
}
}
} // restconf-client-listen-stack-grouping

grouping restconf-client-app-grouping {
    description
        "A grouping for configuring a RESTCONF client
        application that supports both 'initiate' and 'listen'
        protocol stacks for a multiplicity of connections.";
    container initiate {
        if-feature "initiate";
        presence
            "Indicates that client-initiated connections have been
            configured. This statement is present so the mandatory
            descendant nodes do not imply that this node must be
            configured.";
        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
                maintain simultaneous connections with.";
            leaf name {
                type string;
                description
                    "An arbitrary name for the RESTCONF server.";
            }
            container endpoints {
                description
                    "Container for a 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.";
  }
  uses restconf-client-initiate-stack-grouping;
}
}
container connection-type {
  description
    "Indicates the RESTCONF client's preference for how
    the RESTCONF 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 server. If the connection goes down,
          immediately start trying to reconnect to the
          RESTCONF server, 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.";
      }
    }
    case periodic-connection {
      container periodic {
        presence
          "Indicates that a periodic connection is to be
          maintained.";
        description
          "Periodically connect to the RESTCONF server.

          This connection type decreases resource
          utilization, albeit with increased delay
          in RESTCONF server to RESTCONF client
          interactions.
```

The RESTCONF client SHOULD gracefully close the underlying TLS connection upon completing planned activities.

Connections are established at the same start time regardless how long the previous connection stayed open.

In the case that the previous connection is still active, establishing a new connection is NOT RECOMMENDED.";

```
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 '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|1[1-2]'
      + '[0-9]|3[0-1])T(0[0-9]|1[0-9]|2[0-3]):['
      + '0-5][0-9]:00(Z|[\+\-]((1[0-3]|0[0-9]):'
      + '([0-5][0-9])|14:00))?'';
  }
  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.

    If an 'anchor-time' is not provided, then the
    server MAY implicitly set it to the time when
    this configuraton is applied (e.g., on boot).

    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 "180"; // three minutes
  description
    "Specifies the maximum number of seconds
```

```
        that the underlying TCP session may remain
        idle. A TCP 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.";
    }
}
}
}
}
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 known. If
          no previous connection is known, then the
          first endpoint configured is used.";
      }
      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-wait {
      type uint16 {
        range "1..max";
      }
      units "seconds";
      default "5";
      description
        "Specifies the amount of time in seconds after which,
         if the connection is not established, an endpoint
         connection attempt is considered unsuccessful.";
    }
    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).";
    }
  }
} // initiate

container listen {
  if-feature listen;
  presence
    "Indicates that client-listening ports have been configured.
     This statement is present so the mandatory descendant nodes
     do not imply that this node must be configured.";
  description
    "Configures the client to accept call-home TCP connections.";
  leaf idle-timeout {
    type uint16;
    units "seconds";
    default "180";
    description
      "Specifies the maximum number of seconds that an
       underlying TCP session may remain idle. A TCP 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 endpoints {
    description
```

```
        "Container for a list of endpoints.";
    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.";
        }
        uses restconf-client-listen-stack-grouping;
    }
} // listen
} // restconf-client-app-grouping

// Protocol accessible node for servers that implement this module.
container restconf-client {
    if-feature central-restconf-client-supported;
    uses restconf-client-app-grouping;
    description
        "Top-level container for RESTCONF client configuration.";
}

}

<CODE ENDS>
```

3. The "ietf-restconf-server" Module

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

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

3.1. Data Model Overview

This section provides an overview of the "ietf-restconf-server" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-restconf-server" module:

Features:

```
+-- http-listen
+-- https-listen
+-- https-call-home
+-- central-restconf-server-supported
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

3.1.2. Groupings

The "ietf-restconf-server" module defines the following "grouping" statements:

```
* restconf-server-grouping
* restconf-server-listen-stack-grouping
* restconf-server-callhome-stack-grouping
* restconf-server-app-grouping
```

Each of these groupings are presented in the following subsections.

3.1.2.1. The "restconf-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-grouping" grouping:

```
grouping restconf-server-grouping:
  +-- client-identity-mappings
    +---u x509c2n:cert-to-name
```

Comments:

- * The "restconf-server-grouping" defines the configuration for the "RESTCONF" part of a protocol stack. It does not, for instance, define any configuration for the "TCP", "TLS", or "HTTP" protocol layers (for that, see Section 3.1.2.2 and Section 3.1.2.3).
- * The "client-identity-mappings" node defines a mapping from certificate fields to RESTCONF user names.
- * For the referenced grouping statement(s):

- The "cert-to-name" grouping is discussed in Section 4.1 of [RFC7407].

3.1.2.2. The "restconf-server-listen-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-listen-stack-grouping" grouping:

```

grouping restconf-server-listen-stack-grouping:
  +-- (transport)
    +--:(http-over-tcp) {tcp-listen}?
      |  +-- http-over-tcp
      |    +-- external-endpoint!
      |      |  +-- address                inet:host
      |      |  +-- port?                  inet:port-number
      |      |  +-- trusted-proxy-count?   uint8
      |      |  +-- client-cert-var?       string
      |      +-- tcp-server-parameters
      |        |  +---u tcps:tcp-server-grouping
      |        +-- http-server-parameters
      |          |  +---u https:http-server-grouping
      |          +-- restconf-server-parameters
      |            +---u rcs:restconf-server-grouping
    +--:(http-over-tls) {tls-listen}?
      |  +-- http-over-tls
      |    +-- tcp-server-parameters
      |      |  +---u tcps:tcp-server-grouping
      |    +-- tls-server-parameters
      |      |  +---u tlss:tls-server-grouping
      |    +-- http-server-parameters
      |      |  +---u https:http-server-grouping
      |    +-- restconf-server-parameters
      |      +---u rcs:restconf-server-grouping
    +--:(http-over-quic) {quic-listen}?
      |  +-- http-over-quic
      |    +-- udp-server-parameters
      |      |  +---u udps:udp-server
      |    +-- tls-server-parameters
      |      |  +---u tlss:tls-server-grouping
      |    +-- http-server-parameters
      |      |  +---u https:http-server-grouping
      |    +-- restconf-server-parameters
      |      +---u rcs:restconf-server-grouping

```

Comments:

- * The "restconf-server-listen-stack-grouping" defines the configuration for a full RESTCONF protocol stack for RESTCONF servers that listen for connections from RESTCONF clients, as opposed to initiating call-home [RFC8071] connections.
- * The "transport" choice node enables either the HTTP or HTTPS transports to be configured, with each option enabled by a "feature" statement. The HTTP option is provided to support cases where a TLS-terminator is deployed in front of the RESTCONF-server.
- * For the referenced grouping statement(s):
 - The "tcp-server-grouping" grouping is discussed in Section 4.1.2.1 of [RFC9643].
 - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [RFC9645].
 - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
 - The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.3. The "restconf-server-callhome-stack-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-callhome-stack-grouping" grouping:

```
grouping restconf-server-callhome-stack-grouping:
  +-- (transport)
    +--:(http-over-tls) {tls-callhome}?
      +-- http-over-tls
        +-- tcp-client-parameters
          | +---u tcpc:tcp-client-grouping
        +-- tls-server-parameters
          | +---u tlss:tls-server-grouping
        +-- http-server-parameters
          | +---u https:http-server-grouping
        +-- restconf-server-parameters
          +---u rcs:restconf-server-grouping
```

Comments:

- * The "restconf-server-callhome-stack-grouping" defines the configuration for a full RESTCONF protocol stack, for RESTCONF servers that initiate call-home [RFC8071] connections to RESTCONF clients.

- * The "transport" choice node enables transport options to be configured. This document only defines an "https" option, but other options MAY be augmented in.
- * For the referenced grouping statement(s):
 - The "tcp-client-grouping" grouping is discussed in Section 3.1.2.1 of [RFC9643].
 - The "tls-server-grouping" grouping is discussed in Section 4.1.2.1 of [RFC9645].
 - The "http-server-grouping" grouping is discussed in Section 3.1.2.1 of [I-D.ietf-netconf-http-client-server].
 - The "restconf-server-grouping" is discussed in Section 3.1.2.1 of this document.

3.1.2.4. The "restconf-server-app-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "restconf-server-app-grouping" grouping:

```

grouping restconf-server-app-grouping:
  +-- listen! {listen}?
  |   +-- endpoints
  |   |   +-- endpoint* [name]
  |   |   |   +-- name?                                string
  |   |   |   +---u restconf-server-listen-stack-grouping
  |   +-- call-home! {callhome}?
  |   |   +-- restconf-client* [name]
  |   |   |   +-- name?                                string
  |   |   |   +-- endpoints
  |   |   |   |   +-- endpoint* [name]
  |   |   |   |   |   +-- name?                                string
  |   |   |   |   |   +---u restconf-server-callhome-stack-grouping
  |   |   +-- connection-type
  |   |   |   +-- (connection-type)
  |   |   |   |   +--:(persistent-connection)
  |   |   |   |   |   +-- persistent!
  |   |   |   |   +--:(periodic-connection)
  |   |   |   |   |   +-- periodic!
  |   |   |   |   |   |   +-- period?                uint16
  |   |   |   |   |   |   +-- anchor-time?          yang:date-and-time
  |   |   |   |   |   |   +-- idle-timeout?         uint16
  |   |   +-- reconnect-strategy
  |   |   |   +-- start-with?          enumeration
  |   |   |   +-- max-wait?            uint16
  |   |   |   +-- max-attempts?       uint8

```

Comments:

- * The "restconf-server-app-grouping" defines the configuration for a RESTCONF server that supports both listening for connections from RESTCONF clients as well as initiating call-home connections to RESTCONF clients.
- * Both the "listen" and "call-home" subtrees must be enabled by "feature" statements.
- * For the referenced grouping statement(s):
 - The "restconf-server-listen-stack-grouping" grouping is discussed in Section 3.1.2.2 in this document.
 - The "restconf-server-callhome-stack-grouping" grouping is discussed in Section 3.1.2.3 in this document.

3.1.3. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-restconf-server" module:

```
module: ietf-restconf-server
  +--rw restconf-server {central-restconf-server-supported}?
    +---u restconf-server-app-grouping
```

Comments:

- * Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
- * The top-level node "restconf-server" is additionally constrained by the feature "central-restconf-server-supported".
- * The "restconf-server-app-grouping" grouping is discussed in Section 3.1.2.4 in this document.
- * The reason for why "restconf-server-app-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of restconf-server-app-grouping to be instantiated in other locations, as may be needed or desired by some modules.

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 2.2.1 of [RFC9641] and Section 2.2.1 of [RFC9642].

===== NOTE: '\ ' line wrapping per RFC 8792 =====

```
<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>
    <endpoints>
      <endpoint>
        <name>http-over-tls</name>
        <http-over-tls>
          <tcp-server-parameters>
            <local-bind>
              <local-address>192.0.2.2</local-address>
            </local-bind>
          </tcp-server-parameters>
          <tls-server-parameters>
            <server-identity>
              <certificate>
                <central-keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                  <certificate>ex-rsa-cert</certificate>
                </central-keystore-reference>
              </certificate>
            </server-identity>
            <client-authentication>
              <ca-certs>
                <central-truststore-reference>trusted-client-ca-cert\
s</central-truststore-reference>
              </ca-certs>
              <ee-certs>
                <central-truststore-reference>trusted-client-ee-cert\
s</central-truststore-reference>
              </ee-certs>
            </client-authentication>
            <keepalives>
              <peer-allowed-to-send/>
            </keepalives>
          </tls-server-parameters>
          <http-server-parameters>
            <server-name>foo.example.com</server-name>
          </http-server-parameters>
          <restconf-server-parameters>
            <client-identity-mappings>
```

```
    <cert-to-name>
      <id>1</id>
      <fingerprint>11:0A:05:11:00</fingerprint>
      <map-type>x509c2n:specified</map-type>
      <name>scooby-doo</name>
    </cert-to-name>
    <cert-to-name>
      <id>2</id>
      <map-type>x509c2n:san-any</map-type>
    </cert-to-name>
  </client-identity-mappings>
</restconf-server-parameters>
</http-over-tls>
</endpoint>
<endpoint>
  <name>http-over-quic</name>
  <http-over-quic>
    <udp-server-parameters>
      <local-bind>
        <local-address>192.0.2.2</local-address>
      </local-bind>
    </udp-server-parameters>
    <tls-server-parameters>
      <server-identity>
        <certificate>
          <central-keystore-reference>
            <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
            <certificate>ex-rsa-cert</certificate>
          </central-keystore-reference>
        </certificate>
      </server-identity>
      <client-authentication>
        <ca-certs>
          <central-truststore-reference>trusted-client-ca-cert\
s</central-truststore-reference>
        </ca-certs>
        <ee-certs>
          <central-truststore-reference>trusted-client-ee-cert\
s</central-truststore-reference>
        </ee-certs>
      </client-authentication>
      <keepalives>
        <peer-allowed-to-send/>
      </keepalives>
    </tls-server-parameters>
    <http-server-parameters>
      <server-name>foo.example.com</server-name>
    </http-server-parameters>
```

```

    <restconf-server-parameters>
      <client-identity-mappings>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>scooby-doo</name>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
      </client-identity-mappings>
    </restconf-server-parameters>
  </http-over-quic>
</endpoint>
</endpoints>
</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>
        <http-over-tls>
          <tcp-client-parameters>
            <remote-address>east.example.com</remote-address>
            <keepalives>
              <idle-time>7200</idle-time>
              <max-probes>9</max-probes>
              <probe-interval>75</probe-interval>
            </keepalives>
          </tcp-client-parameters>
          <tls-server-parameters>
            <server-identity>
              <certificate>
                <central-keystore-reference>
                  <asymmetric-key>rsa-asymmetric-key</asymmetric-k\
ey>
                </certificate>
                <certificate>ex-rsa-cert</certificate>
              </central-keystore-reference>
            </certificate>
          </server-identity>
          <client-authentication>
            <ca-certs>
              <central-truststore-reference>trusted-client-ca-ce\

```



```
    rts</central-truststore-reference>
      </ca-certs>
      <ee-certs>
        <central-truststore-reference>trusted-client-ee-ce\
rts</central-truststore-reference>
      </ee-certs>
    </client-authentication>
    <keepalives>
      <test-peer-aliveness>
        <max-wait>30</max-wait>
        <max-attempts>3</max-attempts>
      </test-peer-aliveness>
    </keepalives>
  </tls-server-parameters>
  <http-server-parameters>
    <server-name>foo.example.com</server-name>
  </http-server-parameters>
  <restconf-server-parameters>
    <client-identity-mappings>
      <cert-to-name>
        <id>1</id>
        <fingerprint>11:0A:05:11:00</fingerprint>
        <map-type>x509c2n:specified</map-type>
        <name>scooby-doo</name>
      </cert-to-name>
      <cert-to-name>
        <id>2</id>
        <map-type>x509c2n:san-any</map-type>
      </cert-to-name>
    </client-identity-mappings>
  </restconf-server-parameters>
</http-over-tls>
</endpoint>
<endpoint>
  <name>west-data-center</name>
  <http-over-tls>
    <tcp-client-parameters>
      <remote-address>west.example.com</remote-address>
      <keepalives>
        <idle-time>7200</idle-time>
        <max-probes>9</max-probes>
        <probe-interval>75</probe-interval>
      </keepalives>
    </tcp-client-parameters>
    <tls-server-parameters>
      <server-identity>
        <certificate>
          <central-keystore-reference>
```

```

        <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
    </server-identity>
    <client-authentication>
        <ca-certs>
            <central-truststore-reference>trusted-client-ca-certs</central-truststore-reference>
        </ca-certs>
        <ee-certs>
            <central-truststore-reference>trusted-client-ee-certs</central-truststore-reference>
        </ee-certs>
    </client-authentication>
    <keepalives>
        <test-peer-aliveness>
            <max-wait>30</max-wait>
            <max-attempts>3</max-attempts>
        </test-peer-aliveness>
    </keepalives>
</tls-server-parameters>
<http-server-parameters>
    <server-name>foo.example.com</server-name>
</http-server-parameters>
<restconf-server-parameters>
    <client-identity-mappings>
        <cert-to-name>
            <id>1</id>
            <fingerprint>11:0A:05:11:00</fingerprint>
            <map-type>x509c2n:specified</map-type>
            <name>scooby-doo</name>
        </cert-to-name>
        <cert-to-name>
            <id>2</id>
            <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
    </client-identity-mappings>
</restconf-server-parameters>
</http-over-tls>
</endpoint>
</endpoints>
<connection-type>
    <periodic>
        <idle-timeout>300</idle-timeout>
        <anchor-time>2023-03-15T01:30:00Z</anchor-time>
        <period>60</period>
    </periodic>

```

```
        </periodic>
    </connection-type>
    <reconnect-strategy>
        <start-with>last-connected</start-with>
        <max-wait>3</max-wait>
        <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], [RFC9643], [RFC9645], and [I-D.ietf-netconf-http-client-server].

<CODE BEGINS> file "ietf-restconf-server@2025-04-24.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-tcp-client {
    prefix tcpc;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }
}
```

```
import ietf-tcp-server {
  prefix tcps;
  reference
    "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
}

import ietf-tls-server {
  prefix tlss;
  reference
    "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
}

import ietf-http-server {
  prefix https;
  reference
    "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
}

import ietf-udp-server {
  prefix udps;
  reference
    "RFC JJJJ: YANG Groupings for UDP Clients and UDP Servers";
}

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

contact
  "WG Web:  https://datatracker.ietf.org/wg/netconf
  WG List:  NETCONF WG list <mailto:netconf@ietf.org>
  Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

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

  Copyright (c) 2025 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 Revised
  BSD License set forth in Section 4.c of the IETF Trust's
  Legal Provisions Relating to IETF Documents
  (https://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC IIII
  (https://www.rfc-editor.org/info/rfcIIII); see the RFC
```

itself for full legal notices.

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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.";

```
revision 2025-04-24 {
  description
    "Initial version";
  reference
    "RFC IIII: A YANG Data Model for RESTCONF Clients and Server";
}

// Features

feature listen {
  description
    "The 'listen' feature indicates that the RESTCONF server
    supports listening for RESTCONF client connections.";
  reference
    "RFC 8040: RESTCONF Protocol";
}

feature tcp-listen {
  if-feature listen;
  description
    "The 'tcp-listen' feature indicates that the RESTCONF server
    supports opening a port to listen for incoming RESTCONF over
    TCP client connections. The TLS connections are expected to
    be terminated by an external system.";
  reference
    "RFC 8040: RESTCONF Protocol";
}

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 over
    TLS client connections, whereby the TLS connections are
    terminated by the server itself.";
  reference
    "RFC 8040: RESTCONF Protocol";
}
```

```
feature quic-listen {
  if-feature listen;
  description
    "The 'quic-listen' feature indicates that the RESTCONF server
    supports opening a port to listen for incoming RESTCONF over
    QUIC client connections.";
  reference
    "RFC 8040: RESTCONF Protocol";
}

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

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

feature central-restconf-server-supported {
  description
    "The 'central-restconf-server-supported' feature indicates
    that the server supports the top-level 'restconf-server'
    node.

    This feature is needed as some servers may want to use
    features defined in this module, which requires this
    module to be implemented, without having to support
    the top-level 'restconf-server' node.";
}

// Groupings

grouping restconf-server-grouping {
  description
    "A grouping for configuring a RESTCONF server
    without any consideration for how underlying transport
    sessions are established."
```

Note that this grouping uses a fairly typical descendant node name such that a stack of 'uses' statements will have name conflicts. It is intended that the consuming data model will resolve the issue by wrapping the 'uses' statement in a container called, e.g., 'restconf-server-parameters'. This model purposely does not do this itself so as to provide maximum flexibility to consuming models.";

```
container client-identity-mappings {
  description
    "Specifies mappings through which RESTCONF client X.509
    certificates are used to determine 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.";
  uses x509c2n:cert-to-name {
    refine "cert-to-name/fingerprint" {
      mandatory false;
      description
        "A 'fingerprint' value does not need to be specified
        when the 'cert-to-name' mapping is independent of
        fingerprint matching. A 'cert-to-name' having no
        fingerprint value will match any client certificate
        and therefore SHOULD only be present at the end of
        the user-ordered 'cert-to-name' list.";
    }
  }
}

grouping restconf-server-listen-stack-grouping {
  description
    "A grouping for configuring a RESTCONF server
    'listen' protocol stack for listening on a single port.";
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";
    case http-over-tcp {
      if-feature "tcp-listen";
      container http-over-tcp {
        description
          "Configures RESTCONF server stack assuming that
          TLS-termination is handled externally.";
        container external-endpoint {
```

```
presence
  "Identifies that an external endpoint has been
  configured. This statement is present so the
  mandatory descendant nodes do not imply that
  this node must be configured.";
description
  "Provides information for the external endpoint.";
leaf address {
  type inet:host;
  mandatory true;
  description
    "The IP address or hostname of the external
    endpoint clients connect to. This value may
    be used in server-generated messages sent to
    clients.";
}
leaf port {
  type inet:port-number;
  default "443";
  description
    "The port number on the external system clients
    connect to. This value may be used in server
    generated messages sent to clients. If no value
    is specified, the IANA-assigned well-known port
    value for 'https' (443) is assumed.";
}
leaf trusted-proxy-count {
  type uint8 {
    range "1..max"; // non-zero
  }
  default 1;
  description
    "The number of proxies in front of the RESTCONF
    server, including the external endpoint. This
    value may be used to assist parsing proxy-sensitive
    HTTP header fields. If no value is specified, the
    value '1' is assumed.";
}
leaf client-cert-var {
  type string;
  description
    "The HTTP header variable used by the external
    TLS-terminator to relay client certificates to this
    RESTCONF server (e.g., 'X-Client-Cert'). This field
    is not mandatory as RESTCONF client authentication
    does not require client certificates.";
}
}
```



```
    container tcp-server-parameters {
      description
        "TCP-level server parameters.";
      uses tcps:tcp-server-grouping {
        refine "local-bind/local-port" {
          default "80";
          description
            "The RESTCONF server will listen on the IANA-
            assigned well-known port value for 'http'
            (80) if no value is specified.";
        }
      }
    }
  }
  container http-server-parameters {
    description
      "HTTP-level server parameters.";
    uses https:http-server-grouping;
  }
  container restconf-server-parameters {
    description
      "RESTCONF-level server parameters.";
    uses rcs:restconf-server-grouping;
  }
}

case http-over-tls {
  if-feature "tls-listen";
  container http-over-tls {
    description
      "Configures RESTCONF server stack assuming that
      TLS-termination is handled internally (i.e.,
      not by a TLS-terminator in front of the RESTCONF
      server).";
    container tcp-server-parameters {
      description
        "TCP-level server parameters.";
      uses tcps:tcp-server-grouping {
        refine "local-bind/local-port" {
          default "443";
          description
            "The RESTCONF server will listen on the IANA-
            assigned well-known port value for 'https'
            (443) if no value is specified.";
        }
      }
    }
  }
  container tls-server-parameters {
    description
```

```
        "TLS-level server parameters.";
        uses tlss:tls-server-grouping;
    }
    container http-server-parameters {
        description
            "HTTP-level server parameters.";
        uses https:http-server-grouping;
    }
    container restconf-server-parameters {
        description
            "RESTCONF-level server parameters.";
        uses rcs:restconf-server-grouping;
    }
}

case http-over-quic {
    if-feature "quic-listen";
    container http-over-quic {
        description
            "Configures RESTCONF server stack assuming that
            TLS-termination is handled internally (i.e.,
            not by a TLS-terminator in front of the RESTCONF
            server).";
        container udp-server-parameters {
            description
                "UDP-level server parameters.";
            uses udps:udp-server {
                refine "local-bind/local-port" {
                    default "443";
                    description
                        "The RESTCONF server will listen on the IANA-
                        assigned well-known port value for 'https'
                        (443) if no value is specified.";
                }
            }
        }
    }
    container tls-server-parameters {
        description
            "TLS-level server parameters.";
        uses tlss:tls-server-grouping;
    }
    container http-server-parameters {
        description
            "HTTP-level server parameters.";
        uses https:http-server-grouping;
    }
    container restconf-server-parameters {
        description
```

```

        "RESTCONF-level server parameters.";
        uses rcs:restconf-server-grouping;
    }
}
}
}

grouping restconf-server-callhome-stack-grouping {
    description
        "A grouping for configuring a RESTCONF server call home
        protocol stack, for a single outbound connection.";
    choice transport {
        mandatory true;
        description
            "Selects between available transports.";
        case http-over-tls {
            if-feature tls-callhome;
            container http-over-tls {
                description
                    "Configures RESTCONF server stack assuming that
                    TLS-termination is handled internally.";
                container tcp-client-parameters {
                    description
                        "TCP-level client parameters.";
                    uses tcpc:tcp-client-grouping {
                        refine "remote-port" {
                            default "4336";
                            description
                                "The RESTCONF server will attempt to
                                connect to the IANA-assigned well-known
                                port for 'restconf-ch-tls' (4336) if no
                                value is specified.";
                        }
                    }
                }
            }
            container tls-server-parameters {
                description
                    "TLS-level server parameters.";
                uses tlss:tls-server-grouping;
            }
            container http-server-parameters {
                description
                    "HTTP-level server parameters.";
                uses https:http-server-grouping;
            }
            container restconf-server-parameters {
                description

```

```
        "RESTCONF-level server parameters.";
        uses rcs:restconf-server-grouping;
    }
}
}
}

grouping restconf-server-app-grouping {
    description
        "A grouping for configuring a RESTCONF server
        application that supports both 'listen' and call home
        protocol stacks for a multiplicity of connections.";
    container listen {
        if-feature listen;
        presence
            "Identifies that the server has been configured to
            listen for incoming client connections. This statement
            is present so the mandatory descendant nodes do not
            imply that this node must be configured.";
        description
            "Configures the RESTCONF server to listen for RESTCONF
            client connections.";
        container endpoints {
            description
                "Container for a list of endpoints.";
            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.";
                }
                uses restconf-server-listen-stack-grouping;
            }
        }
    } // listen
    container call-home {
        if-feature callhome;
        presence
            "Identifies that the server has been configured to initiate
            call home connections. This statement is present so the
            mandatory descendant nodes do not imply that this node
            must be configured.";
        description
```

```
    "Configures the RESTCONF server to initiate the underlying
      transport connection to RESTCONF clients.";
  list restconf-client {
    key "name";
    min-elements 1;
    description
      "List of RESTCONF clients the RESTCONF server is to
        maintain simultaneous call home connections with.";
    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.";
        }
        uses restconf-server-callhome-stack-grouping;
      }
    }
  }
  container connection-type {
    description
      "Indicates the RESTCONF server's preference for how the
        RESTCONF 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 the RESTCONF client, 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.";

```
}
}
case periodic-connection {
  container periodic {
    presence
      "Indicates that a periodic connection is to be
      maintained.";
    description
      "Periodically connect to the RESTCONF client.

      This connection type decreases resource
      utilization, albeit with increased delay in
      RESTCONF client to RESTCONF server interactions.

      The RESTCONF client SHOULD gracefully close
      the underlying TLS connection upon completing
      planned activities. If the underlying TLS
      connection is not closed gracefully, the
      RESTCONF server MUST immediately attempt
      to reestablish the connection.

      Connections are established at the same start
      time regardless how long the previous connection
      stayed open.

      In the case that the previous connection is
      still active (i.e., the RESTCONF client has not
      closed it yet), establishing a new connection
      is NOT RECOMMENDED.";

    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 '[0-9]{4}-(1[0-2]|0[1-9])-(0[1-9]|[1-2]'
```

```

+ '[0-9]|3[0-1])T(0[0-9]|1[0-9]|2[0-3]):['
+ '0-5][0-9]:00(Z|[\+\-]((1[0-3]|0[0-9])):'
+ '([0-5][0-9])|14:00))?' ;
}
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.

      If an 'anchor-time' is not provided, then the
      server MAY implicitly set it to the time when
      this configuraton is applied (e.g., on boot).

      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 "180";
    description
        "Specifies the maximum number of seconds that
          the underlying TCP session may remain idle.
          A TCP 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 known.  If
            no previous connection is known, then the
            first endpoint configured is used.";
    }
    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-wait {
        type uint16 {
            range "1..max";
        }
        units "seconds";
        default "5";
        description
            "Specifies the amount of time in seconds after which,
            if the connection is not established, an endpoint
            connection attempt is considered unsuccessful.";
    }
    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).";
    }
    }
    }
} // call-home
} // restconf-server-app-grouping
```



```
// Protocol accessible node for servers that implement this module.
container restconf-server {
  if-feature central-restconf-server-supported;
  uses restconf-server-app-grouping;
  description
    "Top-level container for RESTCONF server configuration.";
}
}
```

<CODE ENDS>

4. Security Considerations

4.1. Considerations for the "ietf-restconf-client" YANG Module

This section is modeled after the template defined in Section 3.7.1 of [RFC8407].

The "ietf-restconf-client" YANG module defines data nodes that are designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. The YANG-based management protocols have to use a secure transport layer such as SSH [RFC4252], TLS [RFC8446], or QUIC [RFC9000]. The YANG-based management protocols also have to use mutual authentication.

The Network 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.

This YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

This YANG module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This YANG module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

This YANG module enables the possibility for a RESTCONF client to open a port to listen for non-TLS-based RESTCONF Call Home connections [RFC8071]. Whilst RESTCONF [RFC8040] requires HTTPS, this configuration is provided for deployments having a TLS-terminator (e.g., a load balancer) in front of the TLS client. In such a case, the security boundary extends to the TLS-terminator.

4.2. Considerations for the "ietf-restconf-server" YANG Module

This section is modeled after the template defined in Section 3.7.1 of [RFC8407].

The "ietf-restconf-server" YANG module defines data nodes that are designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. The YANG-based management protocols have to use a secure transport layer such as SSH [RFC4252], TLS [RFC8446], or QUIC [RFC9000]. The YANG-based management protocols also have to use mutual authentication.

The Network 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.

This YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

This YANG module uses groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions.

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This YANG module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

This YANG module enables the possibility for a RESTCONF server to open a port to listen for non-TLS-based RESTCONF connections. Whilst RESTCONF [RFC8040] requires HTTPS, this configuration is provided to support deploying a TLS-terminator (e.g., a load balancer) in front of the TLS server. In such a case, the security boundary extends to the TLS-terminator.

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 IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-server
Registrant Contact: The IESG
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 following registrations are requested:

Name: ietf-restconf-client
Namespace: urn:ietf:params:xml:ns:yang:ietf-restconf-client
Prefix: rcc
Reference: RFC IIII
Maintained by IANA: N

Name: ietf-restconf-server
Namespace: urn:ietf:params:xml:ns:yang:ietf-restconf-server
Prefix: rcs
Reference: RFC IIII
Maintained by IANA: N

6. References

6.1. Normative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-

netconf-http-client-server-25, 12 February 2025,
<<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-25>>.

- [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>>.
- [RFC9642] Watsen, K., "A YANG Data Model for a Keystore", RFC 9642, DOI 10.17487/RFC9642, October 2024, <<https://www.rfc-editor.org/info/rfc9642>>.
- [RFC9643] Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", RFC 9643, DOI 10.17487/RFC9643, October 2024, <<https://www.rfc-editor.org/info/rfc9643>>.

- [RFC9645] Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", RFC 9645, DOI 10.17487/RFC9645, October 2024, <<https://www.rfc-editor.org/info/rfc9645>>.

6.2. Informative References

- [I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-38, 12 February 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-38>>.
- [I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-42, 14 April 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-42>>.
- [I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-12, 12 February 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-12>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <<https://www.rfc-editor.org/info/rfc4252>>.
- [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>>.

- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [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>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.
- [RFC9640] Watsen, K., "YANG Data Types and Groupings for Cryptography", RFC 9640, DOI 10.17487/RFC9640, October 2024, <<https://www.rfc-editor.org/info/rfc9640>>.
- [RFC9641] Watsen, K., "A YANG Data Model for a Truststore", RFC 9641, DOI 10.17487/RFC9641, October 2024, <<https://www.rfc-editor.org/info/rfc9641>>.
- [RFC9644] Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", RFC 9644, DOI 10.17487/RFC9644, October 2024, <<https://www.rfc-editor.org/info/rfc9644>>.

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 accommodate 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

A.9. 08 to 09

- * Corrected use of "mandatory true" for "address" leafs.
- * Updated examples to reflect update to groupings defined in the keystore draft.
- * Updated to use groupings defined in new TCP and HTTP drafts.
- * Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

- * Reformatted YANG modules.

A.11. 10 to 11

- * Adjusted for the top-level "demux container" added to groupings imported from other modules.
- * Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.
- * Updated the boilerplate text in module-level "description" statement to match copyeditor convention.
- * Moved "expanded" tree diagrams to the Appendix.

A.12. 11 to 12

- * Removed the 'must' statement limiting keepalives in periodic connections.
- * Updated models and examples to reflect removal of the "demux" containers in the imported models.
- * Updated the "periodic-connection" description statements to better describe behavior when connections are not closed gracefully.

- * Updated text to better reference where certain examples come from (e.g., which Section in which draft).
- * In the server model, commented out the "must 'pinned-ca-certs or pinned-client-certs'" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.
- * Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

A.13. 12 to 13

- * Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)
- * In ietf-restconf-server, Added 'http-listen' (not https-listen) choice, to support case when server is behind a TLS-terminator.
- * Refactored server module to be more like other 'server' models. If folks like it, will also apply to the client model, as well as to both the netconf client/server models. Now the 'restconf-server-grouping' is just the RC-specific bits (i.e., the "demux" container minus the container), 'restconf-server-[listen|callhome]-stack-grouping' is the protocol stack for a single connection, and 'restconf-server-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

A.14. 13 to 14

- * Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)
- * Adjusting from change in TLS client model (removing the top-level 'certificate' container).
- * Added "external-endpoint" to the "http-listen" choice in ietf-restconf-server.

A.15. 14 to 15

- * Added missing "or https-listen" clause in a "must" expression.

- * Refactored the client module similar to how the server module was refactored in -13. Now the 'restconf-client-grouping' is just the RC-specific bits, the 'restconf-client-[initiate|listen]-stack-grouping' is the protocol stack for a single connection, and 'restconf-client-app-grouping' is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

A.16. 15 to 16

- * Added refinement to make "cert-to-name/fingerprint" be mandatory false.
- * Commented out refinement to "tls-server-grouping/client-authentication" until a better "must" expression is defined.
- * Updated restconf-client example to reflect that http-client-grouping no longer has a "protocol-version" leaf.

A.17. 16 to 17

- * Updated examples to include the "-key-format" nodes.
- * Updated examples to remove the "required" nodes.

A.18. 17 to 18

- * Updated examples to reflect new "bag" addition to truststore.

A.19. 18 to 19

- * Updated examples to remove the 'algorithm' nodes.
- * Updated examples to reflect the new TLS keepalives structure.
- * Removed the 'protocol-versions' node from the restconf-server examples.
- * Added a "Note to Reviewers" note to first page.

A.20. 19 to 20

- * Moved and changed "must" statement so that either TLS *or* HTTP auth must be configured.
- * Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].
- * Updated the Security Considerations section.

A.21. 20 to 21

- * Cleaned up titles in the IANA Considerations section
- * Fixed issues found by the SecDir review of the "keystore" draft.

A.22. 21 to 22

- * Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

A.23. 22 to 23

- * Further clarified why some 'presence' statements are present.
- * Addressed nits found in YANG Doctor reviews.
- * Aligned modules with 'pyang -f' formatting.

A.24. 23 to 24

- * Removed Appendix A with fully-expanded tree diagrams.
- * Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.
- * Minor editorial nits

A.25. 24 to 25

- * Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)
- * Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s)

A.26. 25 to 26

- * Added feature "central-restconf-client-supported" to top-level node "restconf-client".
- * Added feature "central-restconf-server-supported" to top-level node "restconf-server".

A.27. 26 to 27

- * Updated per Shepherd reviews impacting the suite of drafts.
- * Added "max-wait" leaf to the "reconnect-strategy" nodes.

A.28. 27 to 28

- * Updated per Shepherd reviews impacting the suite of drafts.

A.29. 28 to 29

- * Updated (implicitly) via Tom Petch reviews.
- * Fixed pattern statement for "leaf anchor-time".
- * Updated examples to use IETF-sanctioned values.

A.30. 29 to 30

- * Addresses AD review comments.
- * Added note to Editor to fix line foldings.
- * Removed "Conventions" section as there are no "BASE64VALUE=" values used in draft.
- * Removed restconf-client-grouping, since it was empty.
- * Removed erroneous statement "client-identity-mappings" must be enabled by a "feature".
- * Added Security Considerations text to also look a SC-section from imported modules.
- * Removed "A wrapper around the foobar parameters to avoid name collisions" text.
- * Added container "endpoints" to wrap list "endpoint".
- * Fixed if-feature "https-listen" to if-feature "https-call-home".

A.31. 30 to 31

- * Addresses AD review by Rob Wilton.

A.32. 31 to 32

- * Addresses 1st-round of IESG reviews.

A.33. 32 to 34

- * Addresses issues found in OpsDir review of the ssh-client-server draft.

- * s/defines/presents/ in a few places.
- * Add refs to where the 'operational' and 'system' datastores are defined.
- * Renamed Security Considerations section s/Template for/ Considerations for/

A.34. 34 to 36

- * Nothing changed. Only bumped for automation...

A.35. 38 to 39

- * Aligned with updates to 'http-client-server' per IESG review.
- * Renamed some 'feature' statements to be more like http-client-server

A.36. 39 to 40

- * Updated 2nd paragraph in Security Considerations section to match RFC Editor's version for other drafts in the suite of client-server drafts.
- * Added a logically-impossible 'if-feature' statement to the 'uri' components from the 'http-client-grouping', from the 'ietf-http-client' YANG module.

A.37. 40 to 41

- * In both the client and server YANG modules, added "if-feature" statements to dependant features. This oversight was identified by a WG member.
- * In the server YANG module, added "leaf client-cert-var" to the "external-endpoint" container to enable the HTTP header variable used by an external TLS terminator to be configured. The need for this was identified by an ISP deploying a RESTCONF server behind an AWS ALB (application load balancer) that hardcodes an Amazon-specific variable name.

A.38. 41 to 42

- * Updated Security Considerations section per Mike Bishop's review. Now more prominently calls out how HTTP (w/o TLS) may be configured, despite being required by RESTCONF [RFC 8040], to support external TLS-terminators (e.g., load balancers).

A.39. 42 to 43

- * Updated to reflect some of Med's AD-review comments.
- * Fixed Security Considerations section update made in -42.
- * Added 'trusted-proxy-count' to the "external-endpoint" node.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balazs Kovacs, Benoit Claise, Bert Wijnen David Lamparter, Jrgen Schindler, Ladislav Lhotka, Martin Bjrkklund, Med Boucladair, Qiufang Ma, Mehmet Ersue, Michal Vanko, Mike Bishop, Phil Shafer, Qiufang Ma, Radek Krejci, Ramkumar Dhanapal, Rob Wilton, Sean Turner, and Tom Petch.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net