

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
H. Wang  
Huawei  
March 9, 2019

Common YANG Data Types for Cryptography  
draft-ietf-netconf-crypto-types-05

Abstract

This document defines YANG identities, typedefs, the groupings useful for cryptographic applications.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o "XXXX" --> the assigned RFC value for this draft

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix B. 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 September 10, 2019.

## Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. The Crypto Types Module . . . . .	3
2.1. Tree Diagram . . . . .	3
2.2. YANG Module . . . . .	4
3. Security Considerations . . . . .	38
4. IANA Considerations . . . . .	39
4.1. The IETF XML Registry . . . . .	39
4.2. The YANG Module Names Registry . . . . .	39
5. References . . . . .	39
5.1. Normative References . . . . .	39
5.2. Informative References . . . . .	42
Appendix A. Examples . . . . .	44
A.1. The "asymmetric-key-pair-with-certs-grouping" Grouping . . . . .	44
A.2. The "generate-hidden-key" Action . . . . .	46
A.3. The "install-hidden-key" Action . . . . .	47
A.4. The "generate-certificate-signing-request" Action . . . . .	47
A.5. The "certificate-expiration" Notification . . . . .	48
Appendix B. Change Log . . . . .	49
B.1. I-D to 00 . . . . .	49
B.2. 00 to 01 . . . . .	49
B.3. 01 to 02 . . . . .	49
B.4. 02 to 03 . . . . .	50
B.5. 03 to 04 . . . . .	50
B.6. 04 to 05 . . . . .	51
Acknowledgements . . . . .	51
Authors' Addresses . . . . .	51

## 1. Introduction

This document defines a YANG 1.1 [RFC7950] module specifying identities, typedefs, and groupings useful for cryptography.

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

## 2. The Crypto Types Module

### 2.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-crypto-types" module. Only the groupings as represented, as tree diagrams have no means to represent identities or typedefs.

module: ietf-crypto-types

```
grouping public-key-grouping:
  +---- algorithm?      asymmetric-key-algorithm-ref
  +---- public-key?     binary
grouping asymmetric-key-pair-grouping:
  +---- algorithm?      asymmetric-key-algorithm-ref
  +---- public-key?     binary
  +---- private-key?    union
  +---x generate-hidden-key
  |   +---- input
  |   |   +---w algorithm      asymmetric-key-algorithm-ref
  +---x install-hidden-key
  |   +---- input
  |   |   +---w algorithm      asymmetric-key-algorithm-ref
  |   |   +---w public-key?    binary
  |   |   +---w private-key?   binary
grouping trust-anchor-cert-grouping:
  +---- cert?           trust-anchor-cert-cms
  +---n certificate-expiration
  |   +---ro expiration-date   ietf-yang-types:date-and-time
grouping end-entity-cert-grouping:
  +---- cert?           end-entity-cert-cms
  +---n certificate-expiration
  |   +---ro expiration-date   ietf-yang-types:date-and-time
grouping asymmetric-key-pair-with-certs-grouping:
  +---- algorithm?
  |   asymmetric-key-algorithm-ref
  +---- public-key?     binary
```

```

+----- private-key?                                union
+----x generate-hidden-key
|   +----- input
|       +---w algorithm      asymmetric-key-algorithm-ref
+----x install-hidden-key
|   +----- input
|       +---w algorithm      asymmetric-key-algorithm-ref
|       +---w public-key?    binary
|       +---w private-key?   binary
+----- certificates
|   +----- certificate* [name]
|       +----- name                                string
|       +----- cert?                               end-entity-cert-cms
|       +---n certificate-expiration
|           +--ro expiration-date    ietf-yang-types:date-and-time
+----x generate-certificate-signing-request
|   +----- input
|       +---w subject          binary
|       +---w attributes?     binary
+----- output
|       +--ro certificate-signing-request    binary

```

## 2.2. YANG Module

This module has normative references to [RFC2404], [RFC3565], [RFC3686], [RFC4106], [RFC4253], [RFC4279], [RFC4309], [RFC4494], [RFC4543], [RFC4868], [RFC5280], [RFC5652], [RFC5656], [RFC6187], [RFC6991], [RFC7919], [RFC8268], [RFC8332], [RFC8341], [RFC8422], [RFC8446], and [ITU.X690.2015].

This module has an informational reference to [RFC2986], [RFC3174], [RFC4493], [RFC5915], [RFC6125], [RFC6234], [RFC6239], [RFC6507], [RFC8017], [RFC8032], [RFC8439].

<CODE BEGINS> file "ietf-crypto-types@2019-03-09.yang"

```

module ietf-crypto-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-crypto-types";
  prefix ct;

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

```

```
reference
  "RFC 8341: Network Configuration Access Control Model";
}

organization
  "IETF NETCONF (Network Configuration) Working Group";
contact
  "WG Web: <http://datatracker.ietf.org/wg/netconf/>
  WG List: <mailto:netconf@ietf.org>
  Author: Kent Watsen <mailto:kent+ietf@watsen.net>
  Author: Wang Haiguang <wang.haiguang.shieldlab@huawei.com>";
description
  "This module defines common YANG types for cryptographic
  applications.

  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.

  Copyright (c) 2019 IETF Trust and the persons identified
  as authors of the code. All rights reserved.

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

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

revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: Common YANG Data Types for Cryptography";
}

/*****/
/* Identities for Hash Algorithms */
/*****/

identity hash-algorithm {
  description
```

```
    "A base identity for hash algorithm verification.";
}

identity sha-224 {
    base hash-algorithm;
    description
        "The SHA-224 algorithm.";
    reference
        "RFC 6234: US Secure Hash Algorithms.";
}

identity sha-256 {
    base hash-algorithm;
    description
        "The SHA-256 algorithm.";
    reference
        "RFC 6234: US Secure Hash Algorithms.";
}

identity sha-384 {
    base hash-algorithm;
    description
        "The SHA-384 algorithm.";
    reference
        "RFC 6234: US Secure Hash Algorithms.";
}

identity sha-512 {
    base hash-algorithm;
    description
        "The SHA-512 algorithm.";
    reference
        "RFC 6234: US Secure Hash Algorithms.";
}

/*****
/* Identities for Asymmetric Key Algorithms */
*****/

identity asymmetric-key-algorithm {
    description
        "Base identity from which all asymmetric key
        encryption Algorithm.";
}

identity rsa1024 {
    base asymmetric-key-algorithm;
    description
```

```
    "The RSA algorithm using a 1024-bit key.";
  reference
    "RFC 8017:
      PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

identity rsa2048 {
  base asymmetric-key-algorithm;
  description
    "The RSA algorithm using a 2048-bit key.";
  reference
    "RFC 8017:
      PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

identity rsa3072 {
  base asymmetric-key-algorithm;
  description
    "The RSA algorithm using a 3072-bit key.";
  reference
    "RFC 8017:
      PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

identity rsa4096 {
  base asymmetric-key-algorithm;
  description
    "The RSA algorithm using a 4096-bit key.";
  reference
    "RFC 8017:
      PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

identity rsa7680 {
  base asymmetric-key-algorithm;
  description
    "The RSA algorithm using a 7680-bit key.";
  reference
    "RFC 8017:
      PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

identity rsa15360 {
  base asymmetric-key-algorithm;
  description
    "The RSA algorithm using a 15360-bit key.";
  reference
    "RFC 8017:
```

```
        PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }

    identity secp192r1 {
        base asymmetric-key-algorithm;
        description
            "The ECDSA algorithm using a NIST P256 Curve.";
        reference
            "RFC 6090:
            Fundamental Elliptic Curve Cryptography Algorithms.";
    }

    identity secp224r1 {
        base asymmetric-key-algorithm;
        description
            "The ECDSA algorithm using a NIST P256 Curve.";
        reference
            "RFC 6090:
            Fundamental Elliptic Curve Cryptography Algorithms.";
    }

    identity secp256r1 {
        base asymmetric-key-algorithm;
        description
            "The ECDSA algorithm using a NIST P256 Curve.";
        reference
            "RFC 6090:
            Fundamental Elliptic Curve Cryptography Algorithms.";
    }

    identity secp384r1 {
        base asymmetric-key-algorithm;
        description
            "The ECDSA algorithm using a NIST P256 Curve.";
        reference
            "RFC 6090:
            Fundamental Elliptic Curve Cryptography Algorithms.";
    }

    identity secp521r1 {
        base asymmetric-key-algorithm;
        description
            "The ECDSA algorithm using a NIST P256 Curve.";
        reference
            "RFC 6090:
            Fundamental Elliptic Curve Cryptography Algorithms.";
    }
}
```



```

/*****
/*  Identities for MAC Algorithms  */
*****/

identity mac-algorithm {
  description
    "A base identity for mac generation.";
}

identity hmac-sha1 {
  base mac-algorithm;
  description
    "Generating MAC using SHA1 hash function";
  reference
    "RFC 3174: US Secure Hash Algorithm 1 (SHA1)";
}

identity hmac-sha1-96 {
  base mac-algorithm;
  description
    "Generating MAC using SHA1 hash function";
  reference
    "RFC 2404: The Use of HMAC-SHA-1-96 within ESP and AH";
}

identity hmac-sha2-224 {
  base mac-algorithm;
  description
    "Generating MAC using SHA2 hash function";
  reference
    "RFC 6234:
      US Secure Hash Algorithms (SHA and SHA-based HMAC and
      HKDF)";
}

identity hmac-sha2-256 {
  base mac-algorithm;
  description
    "Generating MAC using SHA2 hash function";
  reference
    "RFC 6234:
      US Secure Hash Algorithms (SHA and SHA-based HMAC and
      HKDF)";
}

identity hmac-sha2-256-128 {
  base mac-algorithm;
  description

```

```
        "Generating a 256 bits MAC using SHA2 hash function and
        truncate it to 128 bits";
    reference
        "RFC 4868:
        Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512
        with IPsec";
}

identity hmac-sha2-384 {
    base mac-algorithm;
    description
        "Generating MAC using SHA2 hash function";
    reference
        "RFC 6234:
        US Secure Hash Algorithms (SHA and SHA-based HMAC and
        HKDF)";
}

identity hmac-sha2-384-192 {
    base mac-algorithm;
    description
        "Generating a 384 bits MAC using SHA2 hash function and
        truncate it to 192 bits";
    reference
        "RFC 4868:
        Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with
        IPsec";
}

identity hmac-sha2-512 {
    base mac-algorithm;
    description
        "Generating MAC using SHA2 hash function";
    reference
        "RFC 6234:
        US Secure Hash Algorithms (SHA and SHA-based HMAC and
        HKDF)";
}

identity hmac-sha2-512-256 {
    base mac-algorithm;
    description
        "Generating a 512 bits MAC using SHA2 hash function and
        truncating it to 256 bits";
    reference
        "RFC 4868:
        Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with
        IPsec";
}
```

```
}

identity aes-128-gmac {
  base mac-algorithm;
  description
    "Generating MAC using the Advanced Encryption Standard (AES)
    Galois Message Authentication Code (GMAC) as a mechanism to
    provide data origin authentication";
  reference
    "RFC 4543:
    The Use of Galois Message Authentication Code (GMAC) in
    IPsec ESP and AH";
}

identity aes-192-gmac {
  base mac-algorithm;
  description
    "Generating MAC using the Advanced Encryption Standard (AES)
    Galois Message Authentication Code (GMAC) as a mechanism to
    provide data origin authentication";
  reference
    "RFC 4543:
    The Use of Galois Message Authentication Code (GMAC) in
    IPsec ESP and AH";
}

identity aes-256-gmac {
  base mac-algorithm;
  description
    "Generating MAC using the Advanced Encryption Standard (AES)
    Galois Message Authentication Code (GMAC) as a mechanism to
    provide data origin authentication";
  reference
    "RFC 4543:
    The Use of Galois Message Authentication Code (GMAC) in
    IPsec ESP and AH";
}

identity aes-cmac-96 {
  base mac-algorithm;
  description
    "Generating MAC using Advanced Encryption Standard (AES)
    Cipher-based Message Authentication Code (CMAC)";
  reference
    "RFC 4494: The AES-CMAC-96 Algorithm and its Use with IPsec";
}

identity aes-cmac-128 {
```

```
    base mac-algorithm;
    description
      "Generating MAC using Advanced Encryption Standard (AES)
       Cipher-based Message Authentication Code (CMAC)";
    reference
      "RFC 4493: The AES-CMAC Algorithm";
  }

  /*****
  /* Identities for Encryption Algorithms */
  *****/

  identity encryption-algorithm {
    description
      "A base identity for encryption algorithm.";
  }

  identity aes-128-cbc {
    base encryption-algorithm;
    description
      "Encrypt message with AES algorithm in CBC mode with a key
       length of 128 bits";
    reference
      "RFC 3565:
       Use of the Advanced Encryption Standard (AES) Encryption
       Algorithm in Cryptographic Message Syntax (CMS)";
  }

  identity aes-192-cbc {
    base encryption-algorithm;
    description
      "Encrypt message with AES algorithm in CBC mode with a key
       length of 192 bits";
    reference
      "RFC 3565:
       Use of the Advanced Encryption Standard (AES) Encryption
       Algorithm in Cryptographic Message Syntax (CMS)";
  }

  identity aes-256-cbc {
    base encryption-algorithm;
    description
      "Encrypt message with AES algorithm in CBC mode with a key
       length of 256 bits";
    reference
      "RFC 3565:
       Use of the Advanced Encryption Standard (AES) Encryption
       Algorithm in Cryptographic Message Syntax (CMS)";
  }
```

```
}

identity aes-128-ctr {
  base encryption-algorithm;
  description
    "Encrypt message with AES algorithm in CTR mode with a key
     length of 128 bits";
  reference
    "RFC 3686:
     Using Advanced Encryption Standard (AES) Counter Mode with
     IPsec Encapsulating Security Payload (ESP)";
}

identity aes-192-ctr {
  base encryption-algorithm;
  description
    "Encrypt message with AES algorithm in CTR mode with a key
     length of 192 bits";
  reference
    "RFC 3686:
     Using Advanced Encryption Standard (AES) Counter Mode with
     IPsec Encapsulating Security Payload (ESP)";
}

identity aes-256-ctr {
  base encryption-algorithm;
  description
    "Encrypt message with AES algorithm in CTR mode with a key
     length of 256 bits";
  reference
    "RFC 3686:
     Using Advanced Encryption Standard (AES) Counter Mode with
     IPsec Encapsulating Security Payload (ESP)";
}

/*****
/*  Identities for Encryption and MAC Algorithms  */
*****/

identity encryption-and-mac-algorithm {
  description
    "A base identity for encryption and MAC algorithm.";
}

identity aes-128-ccm {
  base encryption-and-mac-algorithm;
  description
    "Encrypt message with AES algorithm in CCM mode with a key
```

```
        length of 128 bits; it can also be used for generating MAC";
    reference
        "RFC 4309:
        Using Advanced Encryption Standard (AES) CCM Mode with
        IPsec Encapsulating Security Payload (ESP)";
}

identity aes-192-ccm {
    base encryption-and-mac-algorithm;
    description
        "Encrypt message with AES algorithm in CCM mode with a key
        length of 192 bits; it can also be used for generating MAC";
    reference
        "RFC 4309:
        Using Advanced Encryption Standard (AES) CCM Mode with
        IPsec Encapsulating Security Payload (ESP)";
}

identity aes-256-ccm {
    base encryption-and-mac-algorithm;
    description
        "Encrypt message with AES algorithm in CCM mode with a key
        length of 256 bits; it can also be used for generating MAC";
    reference
        "RFC 4309:
        Using Advanced Encryption Standard (AES) CCM Mode with
        IPsec Encapsulating Security Payload (ESP)";
}

identity aes-128-gcm {
    base encryption-and-mac-algorithm;
    description
        "Encrypt message with AES algorithm in GCM mode with a key
        length of 128 bits; it can also be used for generating MAC";
    reference
        "RFC 4106:
        The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
        Security Payload (ESP)";
}

identity aes-192-gcm {
    base encryption-and-mac-algorithm;
    description
        "Encrypt message with AES algorithm in GCM mode with a key
        length of 192 bits; it can also be used for generating MAC";
    reference
        "RFC 4106:
        The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
```

```
        Security Payload (ESP)";
    }

    identity mac-aes-256-gcm {
        base encryption-and-mac-algorithm;
        description
            "Encrypt message with AES algorithm in GCM mode with a key
             length of 128 bits; it can also be used for generating MAC";
        reference
            "RFC 4106:
             The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
             Security Payload (ESP)";
    }

    identity chacha20-poly1305 {
        base encryption-and-mac-algorithm;
        description
            "Encrypt message with chacha20 algorithm and generate MAC with
             POLY1305; it can also be used for generating MAC";
        reference
            "RFC 8439: ChaCha20 and Poly1305 for IETF Protocols";
    }

    /* ***** */
    /* Identities for signature algorithm */
    /* ***** */

    identity signature-algorithm {
        description
            "A base identity for asymmetric key encryption algorithm.";
    }

    identity dsa-sha1 {
        base signature-algorithm;
        description
            "The signature algorithm using DSA algorithm with SHA1 hash
             algorithm";
        reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity rsassa-pkcs1-sha1 {
        base signature-algorithm;
        description
            "The signature algorithm using RSASSA-PKCS1-v1_5 with the SHA1
             hash algorithm.";
        reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }
```

```
}

identity rsassa-pkcs1-sha256 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PKCS1-v1_5 with the
    SHA256 hash algorithm.";
  reference
    "RFC 8332:
      Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell
      (SSH) Protocol
    RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity rsassa-pkcs1-sha384 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PKCS1-v1_5 with the
    SHA384 hash algorithm.";
  reference
    "RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity rsassa-pkcs1-sha512 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PKCS1-v1_5 with the
    SHA512 hash algorithm.";
  reference
    "RFC 8332:
      Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell
      (SSH) Protocol
    RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity rsassa-pss-rsae-sha256 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PSS with mask generation
    function 1 and SHA256 hash algorithm. If the public key is
    carried in an X.509 certificate, it MUST use the rsaEncryption
    OID";
  reference
    "RFC 8446:
      The Transport Layer Security (TLS) Protocol Version 1.3";
```



```
}

identity rsassa-pss-rsae-sha384 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PSS with mask generation
    function 1 and SHA384 hash algorithm. If the public key is
    carried in an X.509 certificate, it MUST use the rsaEncryption
    OID";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity rsassa-pss-rsae-sha512 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PSS with mask generation
    function 1 and SHA512 hash algorithm. If the public key is
    carried in an X.509 certificate, it MUST use the rsaEncryption
    OID";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity rsassa-pss-pss-sha256 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PSS with mask generation
    function 1 and SHA256 hash algorithm. If the public key is
    carried in an X.509 certificate, it MUST use the RSASSA-PSS
    OID";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity rsassa-pss-pss-sha384 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PSS with mask generation
    function 1 and SHA256 hash algorithm. If the public key is
    carried in an X.509 certificate, it MUST use the RSASSA-PSS
    OID";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}
```

```
}

identity rsassa-pss-pss-sha512 {
  base signature-algorithm;
  description
    "The signature algorithm using RSASSA-PSS with mask generation
    function 1 and SHA256 hash algorithm. If the public key is
    carried in an X.509 certificate, it MUST use the RSASSA-PSS
    OID";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity ecdsa-secp256r1-sha256 {
  base signature-algorithm;
  description
    "The signature algorithm using ECDSA with curve name secp256r1
    and SHA256 hash algorithm.";
  reference
    "RFC 5656: Elliptic Curve Algorithm Integration in the
    Secure Shell Transport Layer
    RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity ecdsa-secp384r1-sha384 {
  base signature-algorithm;
  description
    "The signature algorithm using ECDSA with curve name secp384r1
    and SHA384 hash algorithm.";
  reference
    "RFC 5656: Elliptic Curve Algorithm Integration in the
    Secure Shell Transport Layer
    RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity ecdsa-secp521r1-sha512 {
  base signature-algorithm;
  description
    "The signature algorithm using ECDSA with curve name secp521r1
    and SHA512 hash algorithm.";
  reference
    "RFC 5656: Elliptic Curve Algorithm Integration in the
    Secure Shell Transport Layer
    RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}
```

```
}

identity ed25519 {
  base signature-algorithm;
  description
    "The signature algorithm using EdDSA as defined in RFC 8032 or
    its successors.";
  reference
    "RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)";
}

identity ed448 {
  base signature-algorithm;
  description
    "The signature algorithm using EdDSA as defined in RFC 8032 or
    its successors.";
  reference
    "RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)";
}

identity eccsi {
  base signature-algorithm;
  description
    "The signature algorithm using ECCSI signature as defined in
    RFC 6507.";
  reference
    "RFC 6507:
    Elliptic Curve-Based Certificateless Signatures for
    Identity-based Encryption (ECCSI)";
}

/*****
/* Identities for key exchange algorithms */
*****/

identity key-exchange-algorithm {
  description
    "A base identity for Diffie-Hellman based key exchange
    algorithm.";
}

identity psk-only {
  base key-exchange-algorithm;
  description
    "Using Pre-shared key for authentication and key exchange";
  reference
    "RFC 4279:
    Pre-Shared Key cipher suites for Transport Layer Security
```

```
        (TLS)";
    }

    identity dhe-ffdhe2048 {
        base key-exchange-algorithm;
        description
            "Ephemeral Diffie Hellman key exchange with 2048 bit
             finite field";
        reference
            "RFC 7919:
             Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
             for Transport Layer Security (TLS)";
    }

    identity dhe-ffdhe3072 {
        base key-exchange-algorithm;
        description
            "Ephemeral Diffie Hellman key exchange with 3072 bit finite
             field";
        reference
            "RFC 7919:
             Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
             for Transport Layer Security (TLS)";
    }

    identity dhe-ffdhe4096 {
        base key-exchange-algorithm;
        description
            "Ephemeral Diffie Hellman key exchange with 4096 bit
             finite field";
        reference
            "RFC 7919:
             Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
             for Transport Layer Security (TLS)";
    }

    identity dhe-ffdhe6144 {
        base key-exchange-algorithm;
        description
            "Ephemeral Diffie Hellman key exchange with 6144 bit
             finite field";
        reference
            "RFC 7919:
             Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
             for Transport Layer Security (TLS)";
    }

    identity dhe-ffdhe8192 {
```

```
    base key-exchange-algorithm;
    description
      "Ephemeral Diffie Hellman key exchange with 8192 bit
       finite field";
    reference
      "RFC 7919:
       Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
       for Transport Layer Security (TLS)";
  }

  identity psk-dhe-ffdhe2048 {
    base key-exchange-algorithm;
    description
      "Key exchange using pre-shared key with Diffie-Hellman key
       generation mechanism, where the DH group is FFDHE2048";
    reference
      "RFC 8446:
       The Transport Layer Security (TLS) Protocol Version 1.3";
  }

  identity psk-dhe-ffdhe3072 {
    base key-exchange-algorithm;
    description
      "Key exchange using pre-shared key with Diffie-Hellman key
       generation mechanism, where the DH group is FFDHE3072";
    reference
      "RFC 8446:
       The Transport Layer Security (TLS) Protocol Version 1.3";
  }

  identity psk-dhe-ffdhe4096 {
    base key-exchange-algorithm;
    description
      "Key exchange using pre-shared key with Diffie-Hellman key
       generation mechanism, where the DH group is FFDHE4096";
    reference
      "RFC 8446:
       The Transport Layer Security (TLS) Protocol Version 1.3";
  }

  identity psk-dhe-ffdhe6144 {
    base key-exchange-algorithm;
    description
      "Key exchange using pre-shared key with Diffie-Hellman key
       generation mechanism, where the DH group is FFDHE6144";
    reference
      "RFC 8446:
       The Transport Layer Security (TLS) Protocol Version 1.3";
  }
```

```
}

identity psk-dhe-ffdhe8192 {
  base key-exchange-algorithm;
  description
    "Key exchange using pre-shared key with Diffie-Hellman key
    generation mechanism, where the DH group is FFDHE8192";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity ecdhe-secp256r1 {
  base key-exchange-algorithm;
  description
    "Ephemeral Diffie Hellman key exchange with elliptic group
    over curve secp256r1";
  reference
    "RFC 8422:
    Elliptic Curve Cryptography (ECC) Cipher Suites for
    Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity ecdhe-secp384r1 {
  base key-exchange-algorithm;
  description
    "Ephemeral Diffie Hellman key exchange with elliptic group
    over curve secp384r1";
  reference
    "RFC 8422:
    Elliptic Curve Cryptography (ECC) Cipher Suites for
    Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity ecdhe-secp521r1 {
  base key-exchange-algorithm;
  description
    "Ephemeral Diffie Hellman key exchange with elliptic group
    over curve secp521r1";
  reference
    "RFC 8422:
    Elliptic Curve Cryptography (ECC) Cipher Suites for
    Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity ecdhe-x25519 {
  base key-exchange-algorithm;
  description
```

```
    "Ephemeral Diffie Hellman key exchange with elliptic group
    over curve x25519";
  reference
    "RFC 8422:
    Elliptic Curve Cryptography (ECC) Cipher Suites for
    Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity ecdhe-x448 {
  base key-exchange-algorithm;
  description
    "Ephemeral Diffie Hellman key exchange with elliptic group
    over curve x448";
  reference
    "RFC 8422:
    Elliptic Curve Cryptography (ECC) Cipher Suites for
    Transport Layer Security (TLS) Versions 1.2 and Earlier";
}

identity psk-ecdh-secp256r1 {
  base key-exchange-algorithm;
  description
    "Key exchange using pre-shared key with elliptic group-based
    Ephemeral Diffie Hellman key exchange over curve secp256r1";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity psk-ecdh-secp384r1 {
  base key-exchange-algorithm;
  description
    "Key exchange using pre-shared key with elliptic group-based
    Ephemeral Diffie Hellman key exchange over curve secp384r1";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity psk-ecdh-secp521r1 {
  base key-exchange-algorithm;
  description
    "Key exchange using pre-shared key with elliptic group-based
    Ephemeral Diffie Hellman key exchange over curve secp521r1";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}
```

```
identity psk-ecdhe-x25519 {
  base key-exchange-algorithm;
  description
    "Key exchange using pre-shared key with elliptic group-based
    Ephemeral Diffie Hellman key exchange over curve x25519";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity psk-ecdhe-x448 {
  base key-exchange-algorithm;
  description
    "Key exchange using pre-shared key with elliptic group-based
    Ephemeral Diffie Hellman key exchange over curve x448";
  reference
    "RFC 8446:
    The Transport Layer Security (TLS) Protocol Version 1.3";
}

identity diffie-hellman-group14-sha1 {
  base key-exchange-algorithm;
  description
    "Using DH group14 and SHA1 for key exchange";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group14-sha256 {
  base key-exchange-algorithm;
  description
    "Using DH group14 and SHA256 for key exchange";
  reference
    "RFC 8268:
    More Modular Exponentiation (MODP) Diffie-Hellman (DH)
    Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

identity diffie-hellman-group15-sha512 {
  base key-exchange-algorithm;
  description
    "Using DH group15 and SHA512 for key exchange";
  reference
    "RFC 8268:
    More Modular Exponentiation (MODP) Diffie-Hellman (DH)
    Key Exchange (KEX) Groups for Secure Shell (SSH)";
}
```



```
identity diffie-hellman-group16-sha512 {
  base key-exchange-algorithm;
  description
    "Using DH group16 and SHA512 for key exchange";
  reference
    "RFC 8268:
      More Modular Exponentiation (MODP) Diffie-Hellman (DH)
      Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

identity diffie-hellman-group17-sha512 {
  base key-exchange-algorithm;
  description
    "Using DH group17 and SHA512 for key exchange";
  reference
    "RFC 8268:
      More Modular Exponentiation (MODP) Diffie-Hellman (DH)
      Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

identity diffie-hellman-group18-sha512 {
  base key-exchange-algorithm;
  description
    "Using DH group18 and SHA512 for key exchange";
  reference
    "RFC 8268:
      More Modular Exponentiation (MODP) Diffie-Hellman (DH)
      Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

identity ecdh-sha2-secp256r1 {
  base key-exchange-algorithm;
  description
    "Elliptic curve-based Diffie Hellman key exchange over curve
      secp256r1 and using SHA2 for MAC generation";
  reference
    "RFC 6239: Suite B Cryptographic Suites for Secure Shell
      (SSH)";
}

identity ecdh-sha2-secp384r1 {
  base key-exchange-algorithm;
  description
    "Elliptic curve-based Diffie Hellman key exchange over curve
      secp384r1 and using SHA2 for MAC generation";
  reference
    "RFC 6239: Suite B Cryptographic Suites for Secure Shell
      (SSH)";
}
```

```
}

identity rsaes-oaep {
  base key-exchange-algorithm;
  description
    "RSAES-OAEP combines the RSAEP and RSADP primitives with the
     EME-OAEP encoding method";
  reference
    "RFC 8017:
     PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

identity rsaes-pkcs1-v1_5 {
  base key-exchange-algorithm;
  description
    " RSAES-PKCS1-v1_5 combines the RSAEP and RSADP primitives
     with the EME-PKCS1-v1_5 encoding method";
  reference
    "RFC 8017:
     PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

/*****
/*   Typedefs for identityrefs to above base identities   */
*****/

typedef hash-algorithm-ref {
  type identityref {
    base hash-algorithm;
  }
  description
    "This typedef enables importing modules to easily define an
     identityref to the 'hash-algorithm' base identity.";
}

typedef signature-algorithm-ref {
  type identityref {
    base signature-algorithm;
  }
  description
    "This typedef enables importing modules to easily define an
     identityref to the 'signature-algorithm' base identity.";
}

typedef mac-algorithm-ref {
  type identityref {
    base mac-algorithm;
  }
}
```

```
    description
      "This typedef enables importing modules to easily define an
       identityref to the 'mac-algorithm' base identity.";
  }

  typedef encryption-algorithm-ref {
    type identityref {
      base encryption-algorithm;
    }
    description
      "This typedef enables importing modules to easily define an
       identityref to the 'encryption-algorithm'
       base identity.";
  }

  typedef encryption-and-mac-algorithm-ref {
    type identityref {
      base encryption-and-mac-algorithm;
    }
    description
      "This typedef enables importing modules to easily define an
       identityref to the 'encryption-and-mac-algorithm'
       base identity.";
  }

  typedef asymmetric-key-algorithm-ref {
    type identityref {
      base asymmetric-key-algorithm;
    }
    description
      "This typedef enables importing modules to easily define an
       identityref to the 'asymmetric-key-algorithm'
       base identity.";
  }

  typedef key-exchange-algorithm-ref {
    type identityref {
      base key-exchange-algorithm;
    }
    description
      "This typedef enables importing modules to easily define an
       identityref to the 'key-exchange-algorithm' base identity.";
  }

  /*****
  /*  Typedefs for ASN.1 structures from RFC 5280  */
  /*****/
```

```
typedef x509 {
  type binary;
  description
    "A Certificate structure, as specified in RFC 5280,
    encoded using ASN.1 distinguished encoding rules (DER),
    as specified in ITU-T X.690.";
  reference
    "RFC 5280:
      Internet X.509 Public Key Infrastructure Certificate
      and Certificate Revocation List (CRL) Profile
    ITU-T X.690:
      Information technology - ASN.1 encoding rules:
      Specification of Basic Encoding Rules (BER),
      Canonical Encoding Rules (CER) and Distinguished
      Encoding Rules (DER).";
}

typedef crl {
  type binary;
  description
    "A CertificateList structure, as specified in RFC 5280,
    encoded using ASN.1 distinguished encoding rules (DER),
    as specified in ITU-T X.690.";
  reference
    "RFC 5280:
      Internet X.509 Public Key Infrastructure Certificate
      and Certificate Revocation List (CRL) Profile
    ITU-T X.690:
      Information technology - ASN.1 encoding rules:
      Specification of Basic Encoding Rules (BER),
      Canonical Encoding Rules (CER) and Distinguished
      Encoding Rules (DER).";
}

/*****
/*   Typedefs for ASN.1 structures from 5652   */
*****/

typedef cms {
  type binary;
  description
    "A ContentInfo structure, as specified in RFC 5652,
    encoded using ASN.1 distinguished encoding rules (DER),
    as specified in ITU-T X.690.";
  reference
    "RFC 5652:
      Cryptographic Message Syntax (CMS)
    ITU-T X.690:
```

```
        Information technology - ASN.1 encoding rules:
        Specification of Basic Encoding Rules (BER),
        Canonical Encoding Rules (CER) and Distinguished
        Encoding Rules (DER).";
    }

    typedef data-content-cms {
        type cms;
        description
            "A CMS structure whose top-most content type MUST be the
            data content type, as described by Section 4 in RFC 5652.";
        reference
            "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef signed-data-cms {
        type cms;
        description
            "A CMS structure whose top-most content type MUST be the
            signed-data content type, as described by Section 5 in
            RFC 5652.";
        reference
            "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef enveloped-data-cms {
        type cms;
        description
            "A CMS structure whose top-most content type MUST be the
            enveloped-data content type, as described by Section 6
            in RFC 5652.";
        reference
            "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef digested-data-cms {
        type cms;
        description
            "A CMS structure whose top-most content type MUST be the
            digested-data content type, as described by Section 7
            in RFC 5652.";
        reference
            "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

    typedef encrypted-data-cms {
        type cms;
        description
```

```
        "A CMS structure whose top-most content type MUST be the
        encrypted-data content type, as described by Section 8
        in RFC 5652.";
    reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
}

typedef authenticated-data-cms {
    type cms;
    description
        "A CMS structure whose top-most content type MUST be the
        authenticated-data content type, as described by Section 9
        in RFC 5652.";
    reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
}

/*****
/*  Typedefs for structures related to RFC 4253  */
*****/

typedef ssh-host-key {
    type binary;
    description
        "The binary public key data for this SSH key, as
        specified by RFC 4253, Section 6.6, i.e.:

        string    certificate or public key format
                  identifier
        byte[n]    key/certificate data.";
    reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer
        Protocol";
}

/*****
/*  Typedefs for ASN.1 structures related to RFC 5280  */
*****/

typedef trust-anchor-cert-x509 {
    type x509;
    description
        "A Certificate structure that MUST encode a self-signed
        root certificate.";
}

typedef end-entity-cert-x509 {
    type x509;
```

```
description
  "A Certificate structure that MUST encode a certificate
   that is neither self-signed nor having Basic constraint
   CA true.";
}

/*****
/*  Typedefs for ASN.1 structures related to RFC 5652  */
*****/

typedef trust-anchor-cert-cms {
  type signed-data-cms;
  description
    "A CMS SignedData structure that MUST contain the chain of
     X.509 certificates needed to authenticate the certificate
     presented by a client or end-entity.

     The CMS MUST contain only a single chain of certificates.
     The client or end-entity certificate MUST only authenticate
     to last intermediate CA certificate listed in the chain.

     In all cases, the chain MUST include a self-signed root
     certificate. In the case where the root certificate is
     itself the issuer of the client or end-entity certificate,
     only one certificate is present.

     This CMS structure MAY (as applicable where this type is
     used) also contain suitably fresh (as defined by local
     policy) revocation objects with which the device can
     verify the revocation status of the certificates.

     This CMS encodes the degenerate form of the SignedData
     structure that is commonly used to disseminate X.509
     certificates and revocation objects (RFC 5280).";
  reference
    "RFC 5280:
     Internet X.509 Public Key Infrastructure Certificate
     and Certificate Revocation List (CRL) Profile.";
}

typedef end-entity-cert-cms {
  type signed-data-cms;
  description
    "A CMS SignedData structure that MUST contain the end
     entity certificate itself, and MAY contain any number
     of intermediate certificates leading up to a trust
     anchor certificate. The trust anchor certificate
     MAY be included as well."
```

The CMS MUST contain a single end entity certificate.  
The CMS MUST NOT contain any spurious certificates.

This CMS structure MAY (as applicable where this type is used) also contain suitably fresh (as defined by local policy) revocation objects with which the device can verify the revocation status of the certificates.

This CMS encodes the degenerate form of the SignedData structure that is commonly used to disseminate X.509 certificates and revocation objects (RFC 5280).";

reference

"RFC 5280:

Internet X.509 Public Key Infrastructure Certificate  
and Certificate Revocation List (CRL) Profile.";

}

```
/*
 * Groupings for keys and/or certificates
 */
```

grouping public-key-grouping {

description

"A public key.";

leaf algorithm {

type asymmetric-key-algorithm-ref;

description

"Identifies the key's algorithm. More specifically,  
this leaf specifies how the 'public-key' binary leaf  
is encoded.";

reference

"RFC CCCC: Common YANG Data Types for Cryptography";

}

leaf public-key {

type binary;

description

"A binary that contains the value of the public key. The  
interpretation of the content is defined by the key  
algorithm. For example, a DSA key is an integer, an RSA  
key is represented as RSAPublicKey as defined in  
RFC 8017, and an Elliptic Curve Cryptography (ECC) key  
is represented using the 'publicKey' described in  
RFC 5915.";

reference

"RFC 8017: Public-Key Cryptography Standards (PKCS) #1:  
RSA Cryptography Specifications Version 2.2.  
RFC 5915: Elliptic Curve Private Key Structure.";

}



```
}

grouping asymmetric-key-pair-grouping {
  description
    "A private/public key pair.";
  uses public-key-grouping;

  leaf private-key {
    nacm:default-deny-all;
    type union {
      type binary;
      type enumeration {
        enum permanently-hidden {
          description
            "The private key is inaccessible due to being
            protected by the system (e.g., a cryptographic
            hardware module). It is not possible to
            configure a permanently hidden key, as a real
            private key value must be set. Permanently
            hidden keys cannot be archived or backed up.";
        }
      }
    }
  }
  description
    "A binary that contains the value of the private key. The
    interpretation of the content is defined by the key
    algorithm. For example, a DSA key is an integer, an RSA
    key is represented as RSAPrivateKey as defined in
    RFC 8017, and an Elliptic Curve Cryptography (ECC) key
    is represented as ECPrivateKey as defined in RFC 5915.";
  reference
    "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
    RSA Cryptography Specifications Version 2.2.
    RFC 5915: Elliptic Curve Private Key Structure.";
} // private-key

action generate-hidden-key {
  description
    "Requests the device to generate a hidden key using the
    specified asymmetric key algorithm. This action is
    used to request the system to generate a key that
    is 'permanently-hidden', perhaps protected by a
    cryptographic hardware module. The resulting
    asymmetric key values are considered operational
    state and hence present only in <operational>.";
  input {
    leaf algorithm {
      type asymmetric-key-algorithm-ref;
    }
  }
}
```

```
        mandatory true;
        description
            "The algorithm to be used when generating the
             asymmetric key.";
        reference
            "RFC CCCC: Common YANG Data Types for Cryptography";
    }
}
} // generate-hidden-key

action install-hidden-key {
    description
        "Requests the device to load the specified values into
         a hidden key. The resulting asymmetric key values are
         considered operational state and hence present only in
         <operational>.";
    input {
        leaf algorithm {
            type asymmetric-key-algorithm-ref;
            mandatory true;
            description
                "The algorithm to be used when generating the
                 asymmetric key.";
            reference
                "RFC CCCC: Common YANG Data Types for Cryptography";
        }
        leaf public-key {
            type binary;
            description
                "A binary that contains the value of the public key.
                 The interpretation of the content is defined by the key
                 algorithm. For example, a DSA key is an integer, an
                 RSA key is represented as RSAPublicKey as defined in
                 RFC 8017, and an Elliptic Curve Cryptography (ECC) key
                 is represented using the 'publicKey' described in
                 RFC 5915.";
            reference
                "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                 RSA Cryptography Specifications Version 2.2.
                 RFC 5915: Elliptic Curve Private Key Structure.";
        }
        leaf private-key {
            type binary;
            description
                "A binary that contains the value of the private key.
                 The interpretation of the content is defined by the key
                 algorithm. For example, a DSA key is an integer, an RSA
                 key is represented as RSAPrivateKey as defined in
```

```
        RFC 8017, and an Elliptic Curve Cryptography (ECC) key
        is represented as ECPPrivateKey as defined in RFC 5915.";
    reference
        "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
        RSA Cryptography Specifications Version 2.2.
        RFC 5915: Elliptic Curve Private Key Structure.";
    }
}
} // install-hidden-key
} // asymmetric-key-pair-grouping

grouping trust-anchor-cert-grouping {
    description
        "A certificate, and a notification for when it might expire.";
    leaf cert {
        type trust-anchor-cert-cms;
        description
            "The binary certificate data for this certificate.";
        reference
            "RFC YYYY: Common YANG Data Types for Cryptography";
    }
    notification certificate-expiration {
        description
            "A notification indicating that the configured certificate
            is either about to expire or has already expired. When to
            send notifications is an implementation specific decision,
            but it is RECOMMENDED that a notification be sent once a
            month for 3 months, then once a week for four weeks, and
            then once a day thereafter until the issue is resolved.";
        leaf expiration-date {
            type yang:date-and-time;
            mandatory true;
            description
                "Identifies the expiration date on the certificate.";
        }
    }
}

grouping end-entity-cert-grouping {
    description
        "A certificate, and a notification for when it might expire.";
    leaf cert {
        type end-entity-cert-cms;
        description
            "The binary certificate data for this certificate.";
        reference
            "RFC YYYY: Common YANG Data Types for Cryptography";
    }
}
```

```
    }
    notification certificate-expiration {
      description
        "A notification indicating that the configured certificate
        is either about to expire or has already expired. When to
        send notifications is an implementation specific decision,
        but it is RECOMMENDED that a notification be sent once a
        month for 3 months, then once a week for four weeks, and
        then once a day thereafter until the issue is resolved.";
      leaf expiration-date {
        type yang:date-and-time;
        mandatory true;
        description
          "Identifies the expiration date on the certificate.";
      }
    }
  }

  grouping asymmetric-key-pair-with-certs-grouping {
    description
      "A private/public key pair and associated certificates.";
    uses asymmetric-key-pair-grouping;
    container certificates {
      description
        "Certificates associated with this asymmetric key.
        More than one certificate supports, for instance,
        a TPM-protected asymmetric key that has both IDevID
        and LDevID certificates associated.";
      list certificate {
        key "name";
        description
          "A certificate for this asymmetric key.";
        leaf name {
          type string;
          description
            "An arbitrary name for the certificate. If the name
            matches the name of a certificate that exists
            independently in <operational> (i.e., an IDevID),
            then the 'cert' node MUST NOT be configured.";
        }
        uses end-entity-cert-grouping;
      }
    } // certificates

    action generate-certificate-signing-request {
      description
        "Generates a certificate signing request structure for
        the associated asymmetric key using the passed subject
```

and attribute values. The specified assertions need to be appropriate for the certificate's use. For example, an entity certificate for a TLS server SHOULD have values that enable clients to satisfy RFC 6125 processing.";

```
input {
  leaf subject {
    type binary;
    mandatory true;
    description
      "The 'subject' field per the CertificationRequestInfo
      structure as specified by RFC 2986, Section 4.1
      encoded using the ASN.1 distinguished encoding
      rules (DER), as specified in ITU-T X.690.";
    reference
      "RFC 2986:
       PKCS #10: Certification Request Syntax
       Specification Version 1.7.
      ITU-T X.690:
       Information technology - ASN.1 encoding rules:
       Specification of Basic Encoding Rules (BER),
       Canonical Encoding Rules (CER) and Distinguished
       Encoding Rules (DER).";
  }
  leaf attributes {
    type binary;
    description
      "The 'attributes' field from the structure
      CertificationRequestInfo as specified by RFC 2986,
      Section 4.1 encoded using the ASN.1 distinguished
      encoding rules (DER), as specified in ITU-T X.690.";
    reference
      "RFC 2986:
       PKCS #10: Certification Request Syntax
       Specification Version 1.7.
      ITU-T X.690:
       Information technology - ASN.1 encoding rules:
       Specification of Basic Encoding Rules (BER),
       Canonical Encoding Rules (CER) and Distinguished
       Encoding Rules (DER).";
  }
}

output {
  leaf certificate-signing-request {
    type binary;
    mandatory true;
    description
      "A CertificationRequest structure as specified by
```

```
        RFC 2986, Section 4.2 encoded using the ASN.1
        distinguished encoding rules (DER), as specified
        in ITU-T X.690.";
    reference
    "RFC 2986:
        PKCS #10: Certification Request Syntax
        Specification Version 1.7.
    ITU-T X.690:
        Information technology - ASN.1 encoding rules:
        Specification of Basic Encoding Rules (BER),
        Canonical Encoding Rules (CER) and Distinguished
        Encoding Rules (DER).";
    }
}
} // generate-certificate-signing-request
} // asymmetric-key-pair-with-certs-grouping

}

<CODE ENDS>
```

### 3. Security Considerations

In order to use YANG identities for algorithm identifiers, only the most commonly used RSA key lengths are supported for the RSA algorithm. Additional key lengths can be defined in another module or added into a future version of this document.

This document limits the number of elliptical curves supported. This was done to match industry trends and IETF best practice (e.g., matching work being done in TLS 1.3). If additional algorithms are needed, they can be defined by another module or added into a future version of this document.

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

generate-certificate-signing-request: For this action, it is RECOMMENDED that implementations assert channel binding [RFC5056], so as to ensure that the application layer that sent the request is the same as the device authenticated when the secure transport layer was established.

This document uses PKCS #10 [RFC2986] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [RFC4211] was considered, but it was unclear if there

was market demand for it. If it is desired to support CRMF in the future, placing a "choice" statement in both the input and output statements, along with an "if-feature" statement on the CRMF option, would enable a backwards compatible solution.

NACM:default-deny-all is set on asymmetric-key-pair-grouping's "private-key" node, as private keys should never be revealed without explicit permission.

#### 4. IANA Considerations

##### 4.1. The IETF XML Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-crypto-types  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

##### 4.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registration is requested:

name:            ietf-crypto-types  
namespace:      urn:ietf:params:xml:ns:yang:ietf-crypto-types  
prefix:         ct  
reference:       RFC XXXX

#### 5. References

##### 5.1. Normative References

- [ITU.X690.2015]      International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1, August 2015, <<https://www.itu.int/rec/T-REC-X.690/>>.
- [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>>.

- [RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, DOI 10.17487/RFC2404, November 1998, <<https://www.rfc-editor.org/info/rfc2404>>.
- [RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, DOI 10.17487/RFC3565, July 2003, <<https://www.rfc-editor.org/info/rfc3565>>.
- [RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <<https://www.rfc-editor.org/info/rfc3686>>.
- [RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <<https://www.rfc-editor.org/info/rfc4106>>.
- [RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <<https://www.rfc-editor.org/info/rfc4253>>.
- [RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <<https://www.rfc-editor.org/info/rfc4279>>.
- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <<https://www.rfc-editor.org/info/rfc4309>>.
- [RFC4494] Song, JH., Poovendran, R., and J. Lee, "The AES-CMAC-96 Algorithm and Its Use with IPsec", RFC 4494, DOI 10.17487/RFC4494, June 2006, <<https://www.rfc-editor.org/info/rfc4494>>.
- [RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <<https://www.rfc-editor.org/info/rfc4543>>.
- [RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <<https://www.rfc-editor.org/info/rfc4868>>.



- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <<https://www.rfc-editor.org/info/rfc5656>>.
- [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <<https://www.rfc-editor.org/info/rfc6187>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7919] Gillmor, D., "Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)", RFC 7919, DOI 10.17487/RFC7919, August 2016, <<https://www.rfc-editor.org/info/rfc7919>>.
- [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>>.
- [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>>.
- [RFC8268] Baushke, M., "More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)", RFC 8268, DOI 10.17487/RFC8268, December 2017, <<https://www.rfc-editor.org/info/rfc8268>>.
- [RFC8332] Bider, D., "Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol", RFC 8332, DOI 10.17487/RFC8332, March 2018, <<https://www.rfc-editor.org/info/rfc8332>>.

- [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>>.
- [RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <<https://www.rfc-editor.org/info/rfc8422>>.
- [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>>.

## 5.2. Informative References

- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <<https://www.rfc-editor.org/info/rfc2986>>.
- [RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <<https://www.rfc-editor.org/info/rfc3174>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <<https://www.rfc-editor.org/info/rfc4211>>.
- [RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <<https://www.rfc-editor.org/info/rfc4493>>.
- [RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <<https://www.rfc-editor.org/info/rfc5056>>.
- [RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", RFC 5915, DOI 10.17487/RFC5915, June 2010, <<https://www.rfc-editor.org/info/rfc5915>>.

- [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>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.
- [RFC6239] Igoe, K., "Suite B Cryptographic Suites for Secure Shell (SSH)", RFC 6239, DOI 10.17487/RFC6239, May 2011, <<https://www.rfc-editor.org/info/rfc6239>>.
- [RFC6507] Groves, M., "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)", RFC 6507, DOI 10.17487/RFC6507, February 2012, <<https://www.rfc-editor.org/info/rfc6507>>.
- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/info/rfc8032>>.
- [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>>.
- [RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018, <<https://www.rfc-editor.org/info/rfc8439>>.

## Appendix A.   Examples

## A.1.   The "asymmetric-key-pair-with-certs-grouping" Grouping

The following example module has been constructed to illustrate use of the "asymmetric-key-pair-with-certs-grouping" grouping defined in the "ietf-crypto-types" module.

Note that the "asymmetric-key-pair-with-certs-grouping" grouping uses both the "asymmetric-key-pair-grouping" and "end-entity-cert-grouping" groupings, and that the "asymmetric-key-pair-grouping" grouping uses the "public-key-grouping" grouping. Thus, a total of four of the five groupings defined in the "ietf-crypto-types" module are illustrated through the use of this one grouping. The only grouping not represented is the "trust-anchor-cert-grouping" grouping.

```
module ex-crypto-types-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-crypto-types-usage";
  prefix "ectu";

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC XXXX: Common YANG Data Types for Cryptography";
  }

  organization
    "Example Corporation";

  contact
    "Author: YANG Designer <mailto:yang.designer@example.com>";

  description
    "This module illustrates the grouping
    defined in the crypto-types draft called
    'asymmetric-key-pair-with-certs-grouping'.";

  revision "1001-01-01" {
    description
      "Initial version";
    reference
      "RFC ????: Usage Example for RFC XXXX";
  }

  container keys {
    description
      "A container of keys.";
    list key {
      key name;
      leaf name {
        type string;
        description
          "An arbitrary name for this key.";
      }
      uses ct:asymmetric-key-pair-with-certs-grouping;
      description
        "An asymmetric key pair with associated certificates.";
    }
  }
}
```

Given the above example usage module, the following example illustrates some configured keys.

```
<keys xmlns="http://example.com/ns/example-crypto-types-usage">
  <key>
    <name>ex-key</name>
    <algorithm
      xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
      ct:rsa2048
    </algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <certificates>
      <certificate>
        <name>ex-cert</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </key>
</keys>
```

#### A.2. The "generate-hidden-key" Action

The following example illustrates the "generate-hidden-key" action in use with the NETCONF protocol.

REQUEST

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <keys xmlns="http://example.com/ns/example-crypto-types-usage">
      <key>
        <name>empty-key</name>
        <generate-hidden-key>
          <algorithm
            xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
            ct:rsa2048
          </algorithm>
        </generate-hidden-key>
      </key>
    </keys>
  </action>
</rpc>
```

## RESPONSE

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

## A.3. The "install-hidden-key" Action

The following example illustrates the "install-hidden-key" action in use with the NETCONF protocol.

## REQUEST

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <keys xmlns="http://example.com/ns/example-crypto-types-usage">
      <key>
        <name>empty-key</name>
        <install-hidden-key>
          <algorithm
            xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
            ct:rsa2048
          </algorithm>
          <public-key>base64encodedvalue==</public-key>
          <private-key>base64encodedvalue==</private-key>
        </install-hidden-key>
      </key>
    </keys>
  </action>
</rpc>
```

## RESPONSE

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

## A.4. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action in use with the NETCONF protocol.

## REQUEST

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <keys xmlns="http://example.com/ns/example-crypto-types-usage">
      <key>
        <name>ex-key-sect571r1</name>
        <generate-certificate-signing-request>
          <subject>base64encodedvalue==</subject>
          <attributes>base64encodedvalue==</attributes>
        </generate-certificate-signing-request>
      </key>
    </keys>
  </action>
</rpc>
```

## RESPONSE

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <certificate-signing-request
    xmlns="http://example.com/ns/example-crypto-types-usage">
    base64encodedvalue==
  </certificate-signing-request>
</rpc-reply>
```

## A.5. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification in use with the NETCONF protocol.



```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <keys xmlns="http://example.com/ns/example-crypto-types-usage">
    <key>
      <name>locally-defined key</name>
      <certificates>
        <certificate>
          <name>my-cert</name>
          <certificate-expiration>
            <expiration-date>
              2018-08-05T14:18:53-05:00
            </expiration-date>
          </certificate-expiration>
        </certificate>
      </certificates>
    </key>
  </keys>
</notification>
```

## Appendix B. Change Log

### B.1. I-D to 00

- o Removed groupings and notifications.
- o Added typedefs for identityrefs.
- o Added typedefs for other RFC 5280 structures.
- o Added typedefs for other RFC 5652 structures.
- o Added convenience typedefs for RFC 4253, RFC 5280, and RFC 5652.

### B.2. 00 to 01

- o Moved groupings from the draft-ietf-netconf-keystore here.

### B.3. 01 to 02

- o Removed unwanted "mandatory" and "must" statements.
- o Added many new crypto algorithms (thanks Haiguang!)
- o Clarified in asymmetric-key-pair-with-certs-grouping, in certificates/certificate/name/description, that if the name MUST NOT match the name of a certificate that exists independently in

<operational>, enabling certs installed by the manufacturer (e.g., an IDevID).

#### B.4. 02 to 03

- o renamed base identity 'asymmetric-key-encryption-algorithm' to 'asymmetric-key-algorithm'.
- o added new 'asymmetric-key-algorithm' identities for secp192r1, secp224r1, secp256r1, secp384r1, and secp521r1.
- o removed 'mac-algorithm' identities for mac-aes-128-ccm, mac-aes-192-ccm, mac-aes-256-ccm, mac-aes-128-gcm, mac-aes-192-gcm, mac-aes-256-gcm, and mac-chacha20-poly1305.
- o for all -cbc and -ctr identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-algorithm'.
- o for all -ccm and -gcm identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-and-mac-algorithm' and renamed the identity to remove the "enc-" prefix.
- o for all the 'signature-algorithm' based identities, renamed from 'rsa-\*' to 'rsassa-\*'.
- o removed all of the "x509v3-" prefixed 'signature-algorithm' based identities.
- o added 'key-exchange-algorithm' based identities for 'rsaes-oaep' and 'rsaes-pkcs1-v1\_5'.
- o renamed typedef 'symmetric-key-encryption-algorithm-ref' to 'symmetric-key-algorithm-ref'.
- o renamed typedef 'asymmetric-key-encryption-algorithm-ref' to 'asymmetric-key-algorithm-ref'.
- o added typedef 'encryption-and-mac-algorithm-ref'.
- o Updated copyright date, boilerplate template, affiliation, and folding algorithm.

#### B.5. 03 to 04

- o ran YANG module through formatter.

B.6.    04 to 05

- o    fixed broken symlink causing reformatted YANG module to not show.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Martin Bjorklund, Balazs Kovacs, Eric Voit, and Liang Xia.

Authors' Addresses

Kent Watsen  
Watsen Networks

EMail: kent+ietf@watsen.net

Wang Haiguang  
Huawei

EMail: wang.haiguang.shieldlab@huawei.com

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
March 9, 2019

YANG Data Model for a Centralized Keystore Mechanism  
draft-ietf-netconf-keystore-08

Abstract

This document defines a YANG 1.1 module called "ietf-keystore" that enables centralized configuration of asymmetric keys and their associated certificates, and notification for when configured certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o "VVVV" --> the assigned RFC value for this draft

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

Status of This Memo

This Internet-Draft 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 September 10, 2019.

#### Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	3
2. Requirements Language . . . . .	3
3. The Keystore Model . . . . .	4
3.1. Tree Diagram . . . . .	4
3.2. Example Usage . . . . .	6
3.3. YANG Module . . . . .	11
4. Security Considerations . . . . .	16
5. IANA Considerations . . . . .	17
5.1. The IETF XML Registry . . . . .	17
5.2. The YANG Module Names Registry . . . . .	17
6. References . . . . .	17
6.1. Normative References . . . . .	17
6.2. Informative References . . . . .	18
Appendix A. Change Log . . . . .	20
A.1. 00 to 01 . . . . .	20
A.2. 01 to 02 . . . . .	20
A.3. 02 to 03 . . . . .	20
A.4. 03 to 04 . . . . .	20
A.5. 04 to 05 . . . . .	21
A.6. 05 to 06 . . . . .	21
A.7. 06 to 07 . . . . .	21
A.8. 07 to 08 . . . . .	21
Acknowledgements . . . . .	21
Author's Address . . . . .	22

## 1. Introduction

This document defines a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of asymmetric keys and their associated certificates, and notification for when configured certificates are about to expire.

This module also defines Six groupings designed for maximum reuse. These groupings include one for the public half of an asymmetric key, one for both the public and private halves of an asymmetric key, one for both halves of an asymmetric key and a list of associated certificates, one for an asymmetric key that may be configured locally or via a reference to an asymmetric key in the keystore, one for a trust anchor certificate and, lastly, one for an end entity certificate.

Special consideration has been given for systems that have cryptographic hardware, such as a Trusted Protection Module (TPM). These systems are unique in that the cryptographic hardware completely hides the private keys and must perform all private key operations. To support such hardware, the "private-key" can be the special value "permanently-hidden" and the actions "generate-hidden-key" and "generate-certificate-signing-request" can be used to direct these operations to the hardware .

This document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, to support keys and associated certificates installed during manufacturing (e.g., for a IDevID [Std-802.1AR-2009] certificate), it is expected that such data may appear only in <operational>.

While only asymmetric keys are currently supported, the module has been designed to enable other key types to be introduced in the future.

The module does not support protecting the contents of the keystore (e.g., via encryption), though it could be extended to do so in the future.

It is not required that a system has an operating system level keystore utility to implement this module.

## 2. Requirements 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.

### 3. The Keystore Model

#### 3.1. Tree Diagram

This section provides a tree diagrams [RFC8340] for the "ietf-keystore" module that presents both the protocol-accessible "keystore" as well the all the groupings intended for external usage.

```

module: ietf-keystore
  +--rw keystore
    +--rw asymmetric-keys
      +--rw asymmetric-key* [name]
        +--rw name string
        +--rw algorithm?
          | asymmetric-key-algorithm-ref
        +--rw public-key? binary
        +--rw private-key? union
        +---x generate-hidden-key
          | +---w input
          |   +---w algorithm asymmetric-key-algorithm-ref
        +---x install-hidden-key
          | +---w input
          |   +---w algorithm asymmetric-key-algorithm-ref
          |   +---w public-key? binary
          |   +---w private-key? binary
        +--rw certificates
          +--rw certificate* [name]
            +--rw name string
            +--rw cert? end-entity-cert-cms
            +---n certificate-expiration
              +-- expiration-date yang:date-and-time
        +---x generate-certificate-signing-request
          +---w input
          | +---w subject binary
          | +---w attributes? binary
          +--ro output
            +--ro certificate-signing-request binary

grouping local-or-keystore-asymmetric-key-grouping
  +-- (local-or-keystore)
    +--:(local) {local-keys-supported}?
      +-- local-definition
        +-- algorithm? asymmetric-key-algorithm-ref
        +-- public-key? binary
        +-- private-key? union
  
```

```

      +---x generate-hidden-key
      |   +---w input
      |       +---w algorithm      asymmetric-key-algorithm-ref
      +---x install-hidden-key
      |   +---w input
      |       +---w algorithm      asymmetric-key-algorithm-ref
      |       +---w public-key?    binary
      |       +---w private-key?   binary
+---:(keystore) {keystore-supported}?
    +-- keystore-reference?  ks:asymmetric-key-ref
grouping local-or-keystore-asymmetric-key-with-certs-grouping
+-- (local-or-keystore)
+---:(local) {local-keys-supported}?
    +-- local-definition
    +-- algorithm?
    |   asymmetric-key-algorithm-ref
    +-- public-key?          binary
    +-- private-key?         union
    +---x generate-hidden-key
    |   +---w input
    |       +---w algorithm      asymmetric-key-algorithm-ref
    +---x install-hidden-key
    |   +---w input
    |       +---w algorithm      asymmetric-key-algorithm-ref
    |       +---w public-key?    binary
    |       +---w private-key?   binary
    +-- certificates
    |   +-- certificate* [name]
    |       +-- name?            string
    |       +-- cert?            end-entity-cert-cms
    |       +---n certificate-expiration
    |           +-- expiration-date  yang:date-and-time
    +---x generate-certificate-signing-request
    |   +---w input
    |       +---w subject        binary
    |       +---w attributes?    binary
    +--ro output
    |   +--ro certificate-signing-request  binary
+---:(keystore) {keystore-supported}?
    +-- keystore-reference?  ks:asymmetric-key-ref
grouping local-or-keystore-end-entity-cert-with-key-grouping
+-- (local-or-keystore)
+---:(local) {local-keys-supported}?
    +-- local-definition
    +-- algorithm?
    |   asymmetric-key-algorithm-ref
    +-- public-key?          binary
    +-- private-key?         union

```



```

+---x generate-hidden-key
|   +---w input
|       +---w algorithm      asymmetric-key-algorithm-ref
+---x install-hidden-key
|   +---w input
|       +---w algorithm      asymmetric-key-algorithm-ref
|       +---w public-key?    binary
|       +---w private-key?   binary
+-- cert?                    end-entity-cert-cms
+---n certificate-expiration
|   +-- expiration-date      yang:date-and-time
+---:(keystore) {keystore-supported}?
|   +-- keystore-reference?   ks:asymmetric-key-certificate-ref

```

### 3.2. Example Usage

The following example illustrates what a fully configured keystore might look like in <operational>, as described by Section 5.3 in [RFC8342]. This datastore view illustrates data set by the manufacturing process alongside conventional configuration. This keystore instance has four keys, two having one associated certificate, one having two associated certificates, and one empty key.

===== NOTE: '\\\' line wrapping per BCP XX (RFC XXXX) =====

```

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  or:origin="or:intended">
  <asymmetric-keys>

    <asymmetric-key>
      <name>ex-rsa-key</name>
      <algorithm>ct:rsa2048</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
      <certificates>
        <certificate>
          <name>ex-rsa-cert</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
    </asymmetric-key>

    <!-- waiting for Haiguang fix...
    <asymmetric-key>
      <name>tls-ec-key</name>

```

```

    <algorithm>ct:secp256r1</algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <certificates>
      <certificate>
        <name>tls-ec-cert</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </asymmetric-key>
-->

  <asymmetric-key>
    <name>tpm-protected-key</name>
    <algorithm or:origin="or:system">ct:rsa2048</algorithm>
    <private-key or:origin="or:system">permanently-hidden</private\
\key>
    <public-key or:origin="or:system">base64encodedvalue==</public\
\key>
    <certificates>
      <certificate or:origin="or:system">
        <name>builtin-idevid-cert</name>
        <cert or:origin="or:system">base64encodedvalue==</cert>
      </certificate>
      <certificate>
        <name>my-ldevid-cert</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </asymmetric-key>

  <asymmetric-key>
    <name>tpm-protected-key2</name>
    <certificates>
      <certificate>
        <name>builtin-idevid-cert2</name>
      </certificate>
      <certificate>
        <name>my-ldevid-cert2</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </asymmetric-key>

</asymmetric-keys>
</keystore>

```

The following example module has been constructed to illustrate the "local-or-keystore-asymmetric-key-grouping" grouping defined in the "ietf-keystore" module.

```
module ex-keystore-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-keystore-usage";
  prefix "eku";

  import ietf-keystore {
    prefix ks;
    reference
      "RFC VVVV: YANG Data Model for a 'Keystore' Mechanism";
  }

  organization
    "Example Corporation";

  contact
    "Author: YANG Designer <mailto:yang.designer@example.com>";

  description
    "This module illustrates the grouping in the keystore draft called
    'local-or-keystore-asymmetric-key-with-certs-grouping'.";

  revision "YYYY-MM-DD" {
    description
      "Initial version";
    reference
      "RFC XXXX: YANG Data Model for a 'Keystore' Mechanism";
  }

  container keystore-usage {
    description
      "An illustration of the various keystore groupings.";

    list just-a-key {
      key name;
      leaf name {
        type string;
        description
          "An arbitrary name for this key.";
      }
    }
    uses ks:local-or-keystore-asymmetric-key-grouping;
    description
      "An asymmetric key, with no certs, that may be configured
      locally or be a reference to an asymmetric key in the
```

```

        keystore. The intent is to reference just the asymmetric
        key, not any certificates that may also be associated
        with the asymmetric key.";
    }

    list key-with-certs {
        key name;
        leaf name {
            type string;
            description
                "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-with-certs-grouping;
        description
            "An asymmetric key and its associated certs, that may be
            configured locally or be a reference to an asymmetric key
            (and its associated certs) in the keystore.";
    }

    list end-entity-cert-with-key {
        key name;
        leaf name {
            type string;
            description
                "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
        description
            "An end-entity certificate, and its associated private key,
            that may be configured locally or be a reference to a
            specific certificate (and its associated private key) in
            the keystore.";
    }
}
}
}

```

The following example illustrates what two configured keys, one local and the other remote, might look like. This example consistent with other examples above (i.e., the referenced key is in an example above).

```

===== NOTE: '\\\' line wrapping per BCP XX (RFC XXXX) =====
<keystore-usage xmlns="http://example.com/ns/example-keystore-usage">

    <!-- ks:local-or-keystore-asymmetric-key-grouping -->

```

```
<just-a-key>
  <name>a locally-defined key</name>
  <local-definition>
    <algorithm
      xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
      ct:rsa2048
    </algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
  </local-definition>
</just-a-key>

<just-a-key>
  <name>a keystore-defined key (and its associated certs)</name>
  <keystore-reference>ex-rsa-key</keystore-reference>
</just-a-key>

<!-- ks:local-or-keystore-key-and-end-entity-cert-grouping -->

<key-with-certs>
  <name>a locally-defined key with certs</name>
  <local-definition>
    <algorithm
      xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
      ct:rsa2048
    </algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <certificates>
      <certificate>
        <name>a locally-defined cert</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </local-definition>
</key-with-certs>

<key-with-certs>
  <name>a keystore-defined key (and its associated certs)</name>
  <keystore-reference>ex-rsa-key</keystore-reference>
</key-with-certs>

<!-- ks:local-or-keystore-end-entity-cert-with-key-grouping -->

<end-entity-cert-with-key>
  <name>a locally-defined end-entity cert with key</name>
  <local-definition>
    <algorithm
```

```

        xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
        ct:rsa2048
    </algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <cert>base64encodedvalue==</cert>
  </local-definition>
</end-entity-cert-with-key>

<end-entity-cert-with-key>
  <name>a keystore-defined certificate (and its associated key)</n\
\ame>
  <keystore-reference>ex-rsa-cert</keystore-reference>
</end-entity-cert-with-key>

</keystore-usage>

```

### 3.3. YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types], and an informative reference to [RFC8342].

```

<CODE BEGINS> file "ietf-keystore@2019-03-09.yang"
module ietf-keystore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
  prefix ks;

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC CCCC: Common YANG Data Types for Cryptography";
  }

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

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

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

```

`description`

"This module defines a keystore to centralize management of security credentials.

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC VVVV:
      YANG Data Model for a Centralized Keystore Mechanism";
}

// Features

feature keystore-supported {
  description
    "The 'keystore-supported' feature indicates that the server
      supports the keystore.";
}

feature local-keys-supported {
  description
    "The 'local-keys-supported' feature indicates that the
      server supports locally-defined keys.";
}

// Typedefs
```

```
typedef asymmetric-key-ref {
  type leafref {
    path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
      + "/ks:name";
  }
  description
    "This typedef enables modules to easily define a reference
    to an asymmetric key stored in the keystore.";
  reference
    "RFC 8342: Network Management Datastore Architecture (NMDA)";
}

typedef asymmetric-key-certificate-ref {
  type leafref {
    path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
      + "/ks:certificates/ks:certificate/ks:name";
  }
  description
    "This typedef enables modules to easily define a reference
    to a specific certificate associated with an asymmetric key
    stored in the keystore.";
  reference
    "RFC 8342: Network Management Datastore Architecture (NMDA)";
}

// Groupings

grouping local-or-keystore-asymmetric-key-grouping {
  description
    "A grouping that expands to allow the asymmetric key to be
    either stored locally, within the using data model, or be
    a reference to an asymmetric key stored in the keystore.";
  choice local-or-keystore {
    mandatory true;
    case local {
      if-feature "local-keys-supported";
      container local-definition {
        description
          "Container to hold the local key definition.";
        uses ct:asymmetric-key-pair-grouping;
      }
    }
    case keystore {
      if-feature "keystore-supported";
      leaf keystore-reference {
        type ks:asymmetric-key-ref;
        description
          "A reference to an asymmetric key that exists in
```



```
        the keystore. The intent is to reference just the
        asymmetric key, not any certificates that may also
        be associated with the asymmetric key.";
    }
}
description
    "A choice between an inlined definition and a definition
    that exists in the keystore.";
}

grouping local-or-keystore-asymmetric-key-with-certs-grouping {
    description
        "A grouping that expands to allow an asymmetric key and its
        associated certificates to be either stored locally, within
        the using data model, or be a reference to an asymmetric key
        (and its associated certificates) stored in the keystore.";
    choice local-or-keystore {
        mandatory true;
        case local {
            if-feature "local-keys-supported";
            container local-definition {
                description
                    "Container to hold the local key definition.";
                uses ct:asymmetric-key-pair-with-certs-grouping;
            }
        }
        case keystore {
            if-feature "keystore-supported";
            leaf keystore-reference {
                type ks:asymmetric-key-ref;
                description
                    "A reference to a value that exists in the keystore.";
            }
        }
    }
    description
        "A choice between an inlined definition and a definition
        that exists in the keystore.";
}

grouping local-or-keystore-end-entity-cert-with-key-grouping {
    description
        "A grouping that expands to allow an end-entity certificate
        (and its associated private key) to be either stored locally,
        within the using data model, or be a reference to a specific
        certificate in the keystore.";
    choice local-or-keystore {
```

```
mandatory true;
case local {
  if-feature "local-keys-supported";
  container local-definition {
    description
      "Container to hold the local key definition.";
    uses ct:asymmetric-key-pair-grouping;
    uses ct:end-entity-cert-grouping;
  }
}
case keystore {
  if-feature "keystore-supported";
  leaf keystore-reference {
    type ks:asymmetric-key-certificate-ref;
    description
      "A reference to a specific certificate, and its
      associated private key, stored in the keystore.";
  }
}
description
  "A choice between an inlined definition and a definition
  that exists in the keystore.";
}

// protocol accessible nodes

container keystore {
  nacm:default-deny-write;
  description
    "The keystore contains a list of keys.";
  container asymmetric-keys {
    description
      "A list of asymmetric keys.";
    list asymmetric-key {
      must '(algorithm and public-key and private-key)
        or not (algorithm or public-key or private-key)';
      key "name";
      description
        "An asymmetric key.";
      leaf name {
        type string;
        description
          "An arbitrary name for the asymmetric key. If the name
          matches the name of a key that exists independently in
          <operational> (i.e., a 'permanently-hidden' key), then
          the 'algorithm', 'public-key', and 'private-key' nodes
          MUST NOT be configured.";
      }
    }
  }
}
```

```
    }
    uses ct:asymmetric-key-pair-with-certs-grouping;
  }
}
}
}
<CODE ENDS>
```

#### 4. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /: The entire data tree defined by this module is sensitive to write operations. For instance, the addition or removal of keys, certificates, etc., can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for the entire data tree.

- /keystore/asymmetric-keys/asymmetric-key/private-key: When writing this node, implementations MUST ensure that the strength of the key being configured is not greater than the strength of the underlying secure transport connection over which it is communicated. Implementations SHOULD fail the write-request if ever the strength of the private key is greater than the strength of the underlying transport, and alert the client that the strength of the key may have been compromised. Additionally, when deleting this node, implementations SHOULD automatically (without explicit request) zeroize these keys in the most secure manner available, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

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

/keystore/asymmetric-keys/asymmetric-key/private-key: This node is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. The best reason for returning this node is to support backup/restore type workflows. For this reason, the NACM extension "default-deny-all" has been set for this data node. Note that this extension is inherited from the grouping in the [I-D.ietf-netconf-crypto-types] module.

## 5. IANA Considerations

### 5.1. The IETF XML Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-keystore  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

### 5.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registration is requested:

name: ietf-keystore  
namespace: urn:ietf:params:xml:ns:yang:ietf-keystore  
prefix: ks  
reference: RFC VVVV

## 6. References

### 6.1. Normative References

[I-D.ietf-netconf-crypto-types]  
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-02 (work in progress), October 2018.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [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>>.
- [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>>.

## 6.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.

[Std-802.1AR-2009]

Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.

## Appendix A. Change Log

## A.1. 00 to 01

- o Replaced the 'certificate-chain' structures with PKCS#7 structures. (Issue #1)
- o Added 'private-key' as a configurable data node, and removed the 'generate-private-key' and 'load-private-key' actions. (Issue #2)
- o Moved 'user-auth-credentials' to the ietf-ssh-client module. (Issues #4 and #5)

## A.2. 01 to 02

- o Added back 'generate-private-key' action.
- o Removed 'RESTRICTED' enum from the 'private-key' leaf type.
- o Fixed up a few description statements.

## A.3. 02 to 03

- o Changed draft's title.
- o Added missing references.
- o Collapsed sections and levels.
- o Added RFC 8174 to Requirements Language Section.
- o Renamed 'trusted-certificates' to 'pinned-certificates'.
- o Changed 'public-key' from config false to config true.
- o Switched 'host-key' from OneAsymmetricKey to definition from RFC 4253.

## A.4. 03 to 04

- o Added typedefs around leafrefs to common keystore paths
- o Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- o Removed Design Considerations section
- o Moved key and certificate definitions from data tree to groupings

## A.5. 04 to 05

- o Removed trust anchors (now in their own draft)
- o Added back global keystore structure
- o Added groupings enabling keys to either be locally defined or a reference to the keystore.

## A.6. 05 to 06

- o Added feature "local-keys-supported"
- o Added nacm:default-deny-all and nacm:default-deny-write
- o Renamed generate-asymmetric-key to generate-hidden-key
- o Added an install-hidden-key action
- o Moved actions inside fo the "asymmetric-key" container
- o Moved some groupings to draft-ietf-netconf-crypto-types

## A.7. 06 to 07

- o Removed a "require-instance false"
- o Clarified some description statements
- o Improved the keystore-usage examples

## A.8. 07 to 08

- o Added "local-definition" containers to avoid possibility of the action/notification statements being under a "case" statement.
- o Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

## Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Ramkumar Dhanapal, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Mahesh Jethanandani, Radek Krejci, Reshad Rahman, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Eric Voit, Bert Wijnen, and Liang Xia.



Author's Address

Kent Watsen  
Watsen Networks

EMail: [kent+ietf@watsen.net](mailto:kent+ietf@watsen.net)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
March 9, 2019

NETCONF Client and Server Models  
draft-ietf-netconf-netconf-client-server-10

Abstract

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

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

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

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

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

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

#### Status of This Memo

This Internet-Draft 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 September 10, 2019.

#### Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	4
2. Terminology . . . . .	4
3. The NETCONF Client Model . . . . .	4
3.1. Tree Diagram . . . . .	4
3.2. Example Usage . . . . .	13
3.3. YANG Module . . . . .	16
4. The NETCONF Server Model . . . . .	25
4.1. Tree Diagram . . . . .	25
4.2. Example Usage . . . . .	34
4.3. YANG Module . . . . .	39
5. Design Considerations . . . . .	49
5.1. Support all NETCONF transports . . . . .	50
5.2. Enable each transport to select which keys to use . . . . .	50
5.3. Support authenticating NETCONF clients certificates . . . . .	50
5.4. Support mapping authenticated NETCONF client certificates to usernames . . . . .	50
5.5. Support both listening for connections and call home . . . . .	50
5.6. For Call Home connections . . . . .	51
5.6.1. Support more than one NETCONF client . . . . .	51
5.6.2. Support NETCONF clients having more than one endpoint . . . . .	51
5.6.3. Support a reconnection strategy . . . . .	51
5.6.4. Support both persistent and periodic connections . . . . .	51
5.6.5. Reconnection strategy for periodic connections . . . . .	51
5.6.6. Keep-alives for persistent connections . . . . .	52
5.6.7. Customizations for periodic connections . . . . .	52
6. Security Considerations . . . . .	52
7. IANA Considerations . . . . .	53
7.1. The IETF XML Registry . . . . .	53
7.2. The YANG Module Names Registry . . . . .	53
8. References . . . . .	54
8.1. Normative References . . . . .	54
8.2. Informative References . . . . .	55
Appendix A. Change Log . . . . .	56
A.1. 00 to 01 . . . . .	56
A.2. 01 to 02 . . . . .	56
A.3. 02 to 03 . . . . .	56
A.4. 03 to 04 . . . . .	56
A.5. 04 to 05 . . . . .	56
A.6. 05 to 06 . . . . .	57
A.7. 06 to 07 . . . . .	57
A.8. 07 to 08 . . . . .	57
Appendix B. 08 to 09 . . . . .	57
Acknowledgements . . . . .	57
Author's Address . . . . .	58

## 1. Introduction

This document defines two YANG [RFC7950] modules, one module to configure a NETCONF [RFC6241] client and the other module to configure a NETCONF server. Both modules support both NETCONF over SSH [RFC6242] and NETCONF over TLS [RFC7589] and NETCONF Call Home connections [RFC8071].

## 2. Terminology

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

## 3. The NETCONF Client Model

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

This model supports both the SSH and TLS transport protocols, using the SSH client and TLS client groupings defined in [I-D.ietf-netconf-ssh-client-server] and [I-D.ietf-netconf-tls-client-server] respectively.

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

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

### 3.1. Tree Diagram

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

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```
module: ietf-netconf-client
  +--rw netconf-client
    +--rw initiate! {initiate}?
      +--rw netconf-server* [name]
        +--rw name                               string
        +--rw endpoints
```

		+--rw endpoint* [name]	
		+--rw name string	
		+--rw (transport)	
		+++:(ssh) {ssh-initiate}?	
		+--rw ssh	
		+--rw remote-address inet:host	
		+--rw remote-port?	
		inet:port-number	
\ess		+--rw local-address? inet:ip-addr\	
		+--rw local-port?	
		inet:port-number	
\s)?		+--rw tcp-keepalives {tcp-client-keepalive\	
		+--rw idle-time? uint16	
		+--rw max-probes? uint16	
		+--rw probe-interval? uint16	
		+--rw ssh-client-identity	
		+--rw username? string	
		+--rw (auth-type)	
		+++:(password)	
		+--rw password? string	
		+++:(public-key)	
		+--rw public-key	
		+--rw (local-or-keystore)	
		+--:(local)	
\ted)?		{local-keys-support\	
		+--rw local-definition	
		+--rw algorithm?	
\y-algorithm-ref		asymmetric-ke\	
		+--rw public-key?	
		binary	
		+--rw private-key?	
		union	
\-key		+---x generate-hidden\	
		+----w input	
		+----w algorithm	
\ric-key-algorithm-ref		asymmet\	
		+---x install-hidden-\	
\key		+----w input	
		+----w algorithm	
\ric-key-algorithm-ref		asymmet\	

\y?							+---w public-ke\
							binary
\ey?							+---w private-k\
							binary
						+---:(keystore)	{keystore-supporte\
\d)?							+---rw keystore-reference?
							ks:asymmetric-ke\
\y-ref						+---:(certificate)	
						+---rw certificate	{sshcmn:ssh-x509-certs}?
						+---rw (local-or-keystore)	
						+---:(local)	
							{local-keys-suppor\
\ted)?							+---rw local-definition
							+---rw algorithm?
\y-algorithm-ref							asymmetric-ke\
							+---rw public-key?
							binary
						+---rw private-key?	
							union
\-key							+----x generate-hidden\
							+----w input
							+----w algorithm
\ric-key-algorithm-ref							asymmet\
							+----x install-hidden-\
\key							+----w input
							+----w algorithm
\ric-key-algorithm-ref							asymmet\
							+----w public-ke\
\y?							binary
\ey?							+----w private-k\
							binary
						+---rw cert?	
\rt-cms							end-entity-ce\

\iration						+---n certificate-exp\
						+-- expiration-date yang:date-\
\and-time						+--:(keystore) {keystore-supporte\
\d)?						+--rw keystore-reference? ks:asymmetric-ke\
\y-certificate-ref						+--rw ssh-server-auth +--rw pinned-ssh-host-keys?   ta:pinned-host-keys-ref   {ta:ssh-host-keys}? +--rw pinned-ca-certs?   ta:pinned-certificates-ref   {sshcmn:ssh-x509-certs,ta:x509-\
\certificates)?						+--rw pinned-server-certs? ta:pinned-certificates-ref {sshcmn:ssh-x509-certs,ta:x509-\
\certificates)?						+--rw ssh-transport-params   {ssh-client-transport-params-confi\
\g)?						+--rw host-key   +--rw host-key-alg* identityref +--rw key-exchange   +--rw key-exchange-alg* identityref +--rw encryption   +--rw encryption-alg* identityref +--rw mac   +--rw mac-alg* identityref +--rw ssh-keepalives {ssh-client-keepalive\
\s)?						+--rw max-wait? uint16 +--rw max-attempts? uint8 +--:(tls) {tls-initiate)? +--rw tls +--rw remote-address inet:host +--rw remote-port? inet:port-num\
\ber						+--rw local-address? inet:ip-addre\
\ss						+--rw local-port? inet:port-num\
\ber						+--rw tcp-keepalives {tcp-client-keepalive\



[illegible]

```
\d)?
|
|
|                                     +---rw keystore-reference?
|                                     ks:asymmetric-ke\
\y-certificate-ref
|
|
|       +---rw tls-server-auth
|       |   +---rw pinned-ca-certs?
|       |   |       ta:pinned-certificates-ref
|       |   |       {ta:x509-certificates}?
|       |   +---rw pinned-server-certs?
|       |   |       ta:pinned-certificates-ref
|       |   |       {ta:x509-certificates}?
|       +---rw tls-hello-params
|       |       {tls-client-hello-params-config}?
|       +---rw tls-versions
|       |   +---rw tls-version*    identityref
|       +---rw cipher-suites
|       |   +---rw cipher-suite*    identityref
|       +---rw tls-keepalives {tls-client-keepalive\
\s)?
|
|               +---rw max-wait?          uint16
|               +---rw max-attempts?      uint8
+---rw connection-type
|   +---rw (connection-type)
|   |   +---:(persistent-connection)
|   |   |   +---rw persistent!
|   |   +---:(periodic-connection)
|   |   |   +---rw periodic!
|   |   |   +---rw period?              uint16
|   |   |   +---rw anchor-time?        yang:date-and-time
|   |   |   +---rw idle-timeout?       uint16
|   +---rw reconnect-strategy
|   |   +---rw start-with?              enumeration
|   |   +---rw max-attempts?           uint8
+---rw listen! {listen}?
|   +---rw idle-timeout?                uint16
|   +---rw endpoint* [name]
|   |   +---rw name                      string
|   +---rw (transport)
|   |   +---:(ssh) {ssh-listen}?
|   |   |   +---rw ssh
|   |   |   |   +---rw local-address            inet:ip-address
|   |   |   |   +---rw local-port?             inet:port-number
|   |   |   |   +---rw tcp-keepalives {tcp-server-keepalives}?
|   |   |   |   |   +---rw idle-time?          uint16
|   |   |   |   |   +---rw max-probes?         uint16
|   |   |   |   |   +---rw probe-interval?     uint16
|   |   |   +---rw ssh-client-identity
|   |   |   |   +---rw username?                string
```

				<pre> +--rw (auth-type) +--:(password)     +--rw password?      string +--:(public-key)     +--rw public-key         +--rw (local-or-keystore)             +--:(local) {local-keys-supported\ </pre>
\}?				<pre> +--rw local-definition +--rw algorithm?     asymmetric-key-algo\ </pre>
\rithm-ref				<pre> +--rw public-key?     binary +--rw private-key?     union +---x generate-hidden-key     +---w input         +---w algorithm             asymmetric-ke\ </pre>
\y-algorithm-ref				<pre> +---x install-hidden-key +---w input     +---w algorithm         asymmetric-ke\ </pre>
\y-algorithm-ref				<pre> +---w public-key?     binary +---w private-key?     binary +---:(keystore) {keystore-supporte\ </pre>
\d}?				<pre> +--rw keystore-reference?     ks:asymmetric-key-ref +---:(certificate) +--rw certificate {sshcmn:ssh-x509-cert\ </pre>
\s}?				<pre> +--rw (local-or-keystore) +--:(local) {local-keys-supported\ </pre>
\}?				<pre> +--rw local-definition +--rw algorithm?     asymmetric-key-algo\ </pre>
\rithm-ref				<pre> +--rw public-key?     binary +--rw private-key?     union </pre>

\y-algorithm-ref				+---x generate-hidden-key +---w input +---w algorithm asymmetric-ke\
\y-algorithm-ref				+---x install-hidden-key +---w input +---w algorithm   asymmetric-ke\
\me				+---w public-key?   binary +---w private-key?   binary +---rw cert?   end-entity-cert-cms +---n certificate-expiration +--- expiration-date yang:date-and-ti\
\d)?				+---:(keystore) {keystore-supporte\
\ificate-ref				+---rw keystore-reference? ks:asymmetric-key-cert\
\icates)?				+---rw ssh-server-auth +---rw pinned-ssh-host-keys?   ta:pinned-host-keys-ref   {ta:ssh-host-keys}? +---rw pinned-ca-certs?   ta:pinned-certificates-ref   {sshcmn:ssh-x509-certs,ta:x509-certif\
\icates)?				+---rw pinned-server-certs? ta:pinned-certificates-ref {sshcmn:ssh-x509-certs,ta:x509-certif\
\icates)?				+---rw ssh-transport-params {ssh-client-transport-params-config}? +---rw host-key   +---rw host-key-alg* identityref +---rw key-exchange   +---rw key-exchange-alg* identityref +---rw encryption   +---rw encryption-alg* identityref +---rw mac +---rw mac-alg* identityref +---rw ssh-keepalives {ssh-client-keepalives}?

		+--rw max-wait?          uint16
		+--rw max-attempts?     uint8
		+---:(tls) {tls-listen}?
		+--rw tls
		+--rw local-address      inet:ip-address
		+--rw local-port?       inet:port-number
		+--rw tcp-keepalives {tcp-server-keepalives}?
		+--rw idle-time?      uint16
		+--rw max-probes?     uint16
		+--rw probe-interval?  uint16
		+--rw tls-client-identity
		+--rw (auth-type)
		+--:(certificate)
		+--rw certificate
		+--rw (local-or-keystore)
		+--:(local) {local-keys-supported}\
\}?		
		+--rw local-definition
		+--rw algorithm?
\rithm-ref		asymmetric-key-algo\
		+--rw public-key?
		binary
		+--rw private-key?
		union
		+---x generate-hidden-key
		+---w input
		+---w algorithm
\y-algorithm-ref		asymmetric-ke\
		+---x install-hidden-key
		+---w input
		+---w algorithm
\y-algorithm-ref		asymmetric-ke\
		+---w public-key?
		binary
		+---w private-key?
		binary
		+--rw cert?
		end-entity-cert-cms
		+---n certificate-expiration
		+-- expiration-date
\me		yang:date-and-ti\
		+--:(keystore) {keystore-supporte\
\d}?		
		+--rw keystore-reference?

```

\ificate-ref      |
                  | ks:asymmetric-key-cert\
+--rw tls-server-auth
|   +--rw pinned-ca-certs?
|   |   ta:pinned-certificates-ref
|   |   {ta:x509-certificates}?
|   +--rw pinned-server-certs?
|   |   ta:pinned-certificates-ref
|   |   {ta:x509-certificates}?
+--rw tls-hello-params
|   {tls-client-hello-params-config}?
+--rw tls-versions
|   +--rw tls-version*    identityref
+--rw cipher-suites
|   +--rw cipher-suite*   identityref
+--rw tls-keepalives {tls-client-keepalives}?
|   +--rw max-wait?       uint16
|   +--rw max-attempts?   uint8

```

### 3.2. Example Usage

The following example illustrates configuring a NETCONF client to initiate connections, using both the SSH and TLS transport protocols, as well as listening for call-home connections, again using both the SSH and TLS transport protocols.

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

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```

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

  <!-- NETCONF servers to initiate connections to -->
  <initiate>
    <netconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <ssh>
            <remote-address>corp-fw1.example.com</remote-address>
            <tcp-keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </tcp-keepalives>

```

```

    <ssh-client-identity>
      <username>foobar</username>
      <public-key>
        <local-definition>
          <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\
\etf-crypto-types">ct:rsa2048</algorithm>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
        </local-definition>
      </public-key>
    </ssh-client-identity>
    <ssh-server-auth>
      <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
\inned-ca-certs>
      <pinned-server-certs>explicitly-trusted-server-certs</\
\pinned-server-certs>
    </ssh-server-auth>
    <ssh-keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </ssh-keepalives>
  </ssh>
</endpoint>
<endpoint>
  <name>corp-fw2.example.com</name>
  <ssh>
    <remote-address>corp-fw2.example.com</remote-address>
    <tcp-keepalives>
      <idle-time>15</idle-time>
      <max-probes>3</max-probes>
      <probe-interval>30</probe-interval>
    </tcp-keepalives>
    <ssh-client-identity>
      <username>foobar</username>
      <public-key>
        <local-definition>
          <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\
\etf-crypto-types">ct:rsa2048</algorithm>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
        </local-definition>
      </public-key>
    </ssh-client-identity>
    <ssh-server-auth>
      <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
\inned-ca-certs>
      <pinned-server-certs>explicitly-trusted-server-certs</\
\pinned-server-certs>

```

```

        </ssh-server-auth>
        <ssh-keepalives>
            <max-wait>30</max-wait>
            <max-attempts>3</max-attempts>
        </ssh-keepalives>
    </ssh>
</endpoint>
</endpoints>
<connection-type>
    <persistent/>
</connection-type>
<reconnect-strategy>
    <start-with>last-connected</start-with>
</reconnect-strategy>
</netconf-server>
</initiate>

<!-- endpoints to listen for NETCONF Call Home connections on -->
<listen>
    <endpoint>
        <name>Intranet-facing listener</name>
        <ssh>
            <local-address>192.0.2.7</local-address>
            <ssh-client-identity>
                <username>foobar</username>
                <public-key>
                    <local-definition>
                        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
crypto-types">ct:rsa2048</algorithm>
                        <private-key>base64encodedvalue==</private-key>
                        <public-key>base64encodedvalue==</public-key>
                    </local-definition>
                </public-key>
            </ssh-client-identity>
            <ssh-server-auth>
                <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinne\
d-ca-certs>
                <pinned-server-certs>explicitly-trusted-server-certs</pinn\
ed-server-certs>
                <pinned-ssh-host-keys>explicitly-trusted-ssh-host-keys</pi\
nned-ssh-host-keys>
            </ssh-server-auth>
        </ssh>
    </endpoint>
</listen>
</netconf-client>

```



### 3.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7589], [RFC8071], [I-D.kwatsen-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

```
<CODE BEGINS> file "ietf-netconf-client@2019-03-09.yang"
module ietf-netconf-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix ncc;

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

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

  import ietf-ssh-client {
    prefix sshc;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC YYYY: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-client {
    prefix tlsc;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
```

```
"WG Web:    <http://datatracker.ietf.org/wg/netconf/>
WG List:    <mailto:netconf@ietf.org>
Author:     Kent Watsen <mailto:kent+ietf@watsen.net>
Author:     Gary Wu <mailto:garywu@cisco.com>";
```

description

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

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Client and Server Models";
}

// Features

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

feature ssh-initiate {
  description
    "The 'ssh-initiate' feature indicates that the NETCONF client
```

```
        supports initiating SSH connections to NETCONF servers.";
    reference
        "RFC 6242:
        Using the NETCONF Protocol over Secure Shell (SSH)";
}

feature tls-initiate {
    description
        "The 'tls-initiate' feature indicates that the NETCONF client
        supports initiating TLS connections to NETCONF servers.";
    reference
        "RFC 7589: Using the NETCONF Protocol over Transport
        Layer Security (TLS) with Mutual X.509 Authentication";
}

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

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

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

// Groupings

grouping netconf-client-grouping {
    description
        "Top-level grouping for NETCONF client configuration.";
    container initiate {
        if-feature "initiate";
    }
}
```

```
presence "Enables client to initiate TCP connections";
description
  "Configures client initiating underlying TCP connections.";
list netconf-server {
  key "name";
  min-elements 1;
  description
    "List of NETCONF servers the NETCONF client is to
    initiate connections to in parallel.";
  leaf name {
    type string;
    description
      "An arbitrary name for the NETCONF server.";
  }
  container endpoints {
    description
      "Container for the list of endpoints.";
    list endpoint {
      key "name";
      min-elements 1;
      ordered-by user;
      description
        "A user-ordered list of endpoints that the NETCONF
        client will attempt to connect to in the specified
        sequence. Defining more than one enables
        high-availability.";
      leaf name {
        type string;
        description
          "An arbitrary name for the endpoint.";
      }
    }
    choice transport {
      mandatory true;
      description
        "Selects between available transports.";
      case ssh {
        if-feature "ssh-initiate";
        container ssh {
          description
            "Specifies IP and SSH specific configuration
            for the connection.";
          uses tcpc:tcp-client-grouping {
            refine "remote-port" {
              default "830";
              description
                "The NETCONF client will attempt to connect
                to the IANA-assigned well-known port value
                for 'netconf-ssh' (443) if no value is
```

```
        specified.";
    }
}
uses sshc:ssh-client-grouping;
}
}
case tls {
  if-feature "tls-initiate";
  container tls {
    description
      "Specifies IP and TLS specific configuration
       for the connection.";
    uses tcpc:tcp-client-grouping {
      refine "remote-port" {
        default "6513";
        description
          "The NETCONF client will attempt to connect
           to the IANA-assigned well-known port value
           for 'netconf-tls' (6513) if no value is
           specified.";
      }
    }
    uses tlsc:tls-client-grouping {
      refine "tls-client-identity/auth-type" {
        mandatory true;
        description
          "NETCONF/TLS clients MUST pass some
           authentication credentials.";
      }
    }
  }
}
} // choice transport
} // list endpoint
} // container endpoints

container connection-type {
  description
    "Indicates the NETCONF client's preference for how the
     NETCONF 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 NETCONF
  server. If the connection goes down, immediately
  start trying to reconnect to it, using the
  reconnection strategy.

  This connection type minimizes any NETCONF server
  to NETCONF 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 NETCONF server. The
      NETCONF server should close the connection upon
      completing planned activities.

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

```

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

```
        the NETCONF client should start with when trying
        to connect to the NETCONF server.";
    }
    leaf max-attempts {
        type uint8 {
            range "1..max";
        }
        default "3";
        description
            "Specifies the number times the NETCONF client tries
            to connect to a specific endpoint before moving on
            to the next endpoint in the list (round robin).";
    }
}
} // netconf-server
} // initiate

container listen {
    if-feature "listen";
    presence "Enables client to accept call-home connections";
    description
        "Configures client accepting call-home TCP connections.";
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default "3600"; // one hour
        description
            "Specifies the maximum number of seconds that a NETCONF
            session may remain idle. A NETCONF session will be
            dropped if it is idle for an interval longer than this
            number of seconds. If set to zero, then the server
            will never drop a session because it is idle. Sessions
            that have a notification subscription active are never
            dropped.";
    }
}
list endpoint {
    key "name";
    min-elements 1;
    description
        "List of endpoints to listen for NETCONF connections.";
    leaf name {
        type string;
        description
            "An arbitrary name for the NETCONF listen endpoint.";
    }
}
choice transport {
    mandatory true;
    description
```



```
    "Selects between available transports.";
case ssh {
  if-feature "ssh-listen";
  container ssh {
    description
      "SSH-specific listening configuration for inbound
      connections.";
    uses tcps:tcp-server-grouping {
      refine "local-port" {
        default "4334";
        description
          "The NETCONF client will listen on the IANA-
          assigned well-known port for 'netconf-ch-ssh'
          (4334) if no value is specified.";
      }
    }
    uses sshc:ssh-client-grouping;
  }
}
case tls {
  if-feature "tls-listen";
  container tls {
    description
      "TLS-specific listening configuration for inbound
      connections.";
    uses tcps:tcp-server-grouping {
      refine "local-port" {
        default "4334";
        description
          "The NETCONF client will listen on the IANA-
          assigned well-known port for 'netconf-ch-ssh'
          (4334) if no value is specified.";
      }
    }
    uses tlsc:tls-client-grouping {
      refine "tls-client-identity/auth-type" {
        mandatory true;
        description
          "NETCONF/TLS clients MUST pass some
          authentication credentials.";
      }
    }
  }
}
} // transport
} // endpoint
} // listen
} // netconf-client
```

```

// Protocol accessible node, for servers that implement this
// module.

container netconf-client {
  uses netconf-client-grouping;
  description
    "Top-level container for NETCONF client configuration.";
}
}
<CODE ENDS>

```

#### 4. The NETCONF Server Model

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

This model supports both the SSH and TLS transport protocols, using the SSH server and TLS server groupings defined in [I-D.ietf-netconf-ssh-client-server] and [I-D.ietf-netconf-tls-client-server] respectively.

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

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

##### 4.1. Tree Diagram

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

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```

module: ietf-netconf-server
  +--rw netconf-server
    +--rw listen! {listen}?
      +--rw idle-timeout?   uint16
      +--rw endpoint* [name]
        +--rw name          string
        +--rw (transport)
          +--:(ssh) {ssh-listen}?
            +--rw ssh
              +--rw local-address      inet:ip-address
              +--rw local-port?        inet:port-number

```

					<pre> +--rw tcp-keepalives {tcp-server-keepalives}?     +--rw idle-time?          uint16     +--rw max-probes?         uint16     +--rw probe-interval?    uint16 +--rw ssh-server-identity     +--rw host-key* [name]         +--rw name                string         +--rw (host-key-type)             +--:(public-key)                 +--rw public-key                     +--rw (local-or-keystore)                         +--:(local)                             {local-keys-supported\ </pre>
\}?					<pre> +--rw local-definition +--rw algorithm?     asymmetric-key-a\ </pre>
\lgorithm-ref					<pre> +--rw public-key?     binary +--rw private-key?     union +---x generate-hidden-key     +---w input         +---w algorithm             asymmetric\ </pre>
\-key-algorithm-ref					<pre> +---x install-hidden-key +---w input     +---w algorithm         asymmetric\ </pre>
\-key-algorithm-ref					<pre> +---w public-key?     binary +---w private-key?     binary +--:(keystore)     {keystore-supported}? +--rw keystore-reference?     ks:asymmetric-key-r\ </pre>
\ef					<pre> +--:(certificate) +--rw certificate     {sshcmn:ssh-x509-certs}? +--rw (local-or-keystore) +--:(local)     {local-keys-supported\ </pre>
\}?					

				<pre> +--rw local-definition +--rw algorithm?     asymmetric-key-a\ </pre>
\algorithm-ref				<pre> +--rw public-key?     binary +--rw private-key?     union +---x generate-hidden-key     +---w input         +---w algorithm             asymmetric\ </pre>
\-key-algorithm-ref				<pre> +---x install-hidden-key     +---w input         +---w algorithm             asymmetric\ </pre>
\-key-algorithm-ref				<pre>         +---w public-key?             binary         +---w private-key?             binary +--rw cert?     end-entity-cert-\ </pre>
\cms				
\tion				<pre> +---n certificate-expira\ </pre>
\-time				<pre> +-- expiration-date     yang:date-and\ </pre>
				<pre> +---: (keystore)     {keystore-supported}? +--rw keystore-reference?     ks:asymmetric-key-c\ </pre>
\ertificate-ref				
\s)?				<pre> +--rw ssh-client-cert-auth {sshcmn:ssh-x509-cert\ </pre>
				<pre> +--rw pinned-ca-certs?     ta:pinned-certificates-ref         {ta:x509-certificates}? +--rw pinned-client-certs?     ta:pinned-certificates-ref         {ta:x509-certificates}? +--rw ssh-transport-params     {ssh-server-transport-params-config}? +--rw host-key     +--rw host-key-alg*   identityref +--rw key-exchange </pre>

```

| | | | | +---rw key-exchange-alg* identityref
| | | | | +---rw encryption
| | | | | | +---rw encryption-alg* identityref
| | | | | +---rw mac
| | | | | | +---rw mac-alg* identityref
| | | | | +---rw ssh-keepalives {ssh-server-keepalives}?
| | | | | +---rw max-wait? uint16
| | | | | +---rw max-attempts? uint8
+---:(tls) {tls-listen}?
+---rw tls
+---rw local-address inet:ip-address
+---rw local-port? inet:port-number
+---rw tcp-keepalives {tcp-server-keepalives}?
| +---rw idle-time? uint16
| +---rw max-probes? uint16
| +---rw probe-interval? uint16
+---rw tls-server-identity
+---rw (local-or-keystore)
+---:(local) {local-keys-supported}?
| | +---rw local-definition
| | | +---rw algorithm?
| | | | asymmetric-key-algorithm-ref
| | | +---rw public-key? bina\
\ry
| | | | +---rw private-key? union
| | | | +---x generate-hidden-key
| | | | | +---w input
| | | | | +---w algorithm
| | | | | asymmetric-key-algorit\
\hm-ref
| | | | +---x install-hidden-key
| | | | | +---w input
| | | | | +---w algorithm
| | | | | asymmetric-key-algorit\
\hm-ref
| | | | | +---w public-key? binary
| | | | | +---w private-key? binary
| | | | +---rw cert?
| | | | | end-entity-cert-cms
| | | | +---n certificate-expiration
| | | | | expiration-date
| | | | | yang:date-and-time
+---:(keystore) {keystore-supported}?
+---rw keystore-reference?
ks:asymmetric-key-certificate-r\
\ef
+---rw tls-client-auth
| +---rw pinned-ca-certs?
```

```

|         |         |         |         ta:pinned-certificates-ref
|         |         |         |         {ta:x509-certificates}?
|         |         |         +---rw pinned-client-certs?
|         |         |         |         ta:pinned-certificates-ref
|         |         |         |         {ta:x509-certificates}?
|         |         |         +---rw cert-maps
|         |         |         |         +---rw cert-to-name* [id]
|         |         |         |         |         +---rw id                               uint32
|         |         |         |         |         +---rw fingerprint
|         |         |         |         |         |         x509c2n:tls-fingerprint
|         |         |         |         |         +---rw map-type                       identityref
|         |         |         |         |         +---rw name                         string
|         |         |         +---rw tls-hello-params
|         |         |         |         {tls-server-hello-params-config}?
|         |         |         +---rw tls-versions
|         |         |         |         +---rw tls-version*   identityref
|         |         |         +---rw cipher-suites
|         |         |         |         +---rw cipher-suite*   identityref
|         |         |         +---rw tls-keepalives {tls-server-keepalives}?
|         |         |         |         +---rw max-wait?       uint16
|         |         |         |         +---rw max-attempts?    uint8
+---rw call-home! {call-home}?
|     +---rw netconf-client* [name]
|     |         +---rw name                             string
|     |         +---rw endpoints
|     |         |         +---rw endpoint* [name]
|     |         |         |         +---rw name                 string
|     |         |         |         +---rw (transport)
|     |         |         |         |         +---:(ssh) {ssh-call-home}?
|     |         |         |         |         |         +---rw ssh
|     |         |         |         |         |         |         +---rw remote-address           inet:host
|     |         |         |         |         |         |         +---rw remote-port?
|     |         |         |         |         |         |         |         inet:port-number
|     |         |         |         |         |         +---rw local-address?             inet:ip-addr\
\ess
|     |         |         |         |         +---rw local-port?
|     |         |         |         |         |         inet:port-number
|     |         |         |         +---rw tcp-keepalives {tcp-client-keepalive\
\s}?
|     |         |         |         |         +---rw idle-time?               uint16
|     |         |         |         |         +---rw max-probes?              uint16
|     |         |         |         |         +---rw probe-interval?          uint16
|     |         |         +---rw ssh-server-identity
|     |         |         |         +---rw host-key* [name]
|     |         |         |         |         +---rw name                             string
|     |         |         |         |         +---rw (host-key-type)
|     |         |         |         |         |         +---:(public-key)
|     |         |         |         |         |         |         +---rw public-key

```

						<pre> +--rw (local-or-keystore) +--:(local)   {local-keys-sup\ </pre>
\ported}?						
						<pre> +--rw local-definition +--rw algorithm?   asymmetric\ </pre>
\-key-algorithm-ref						
						<pre> +--rw public-key?   binary +--rw private-key?   union +---x generate-hid\ </pre>
\den-key						
						<pre>   +---w input +---w algori\ </pre>
\thm						
						<pre>   asym\ </pre>
\metric-key-algorithm-ref						
						<pre> +---x install-hidd\ </pre>
\en-key						
						<pre> +---w input +---w algori\ </pre>
\thm						
						<pre>   asym\ </pre>
\metric-key-algorithm-ref						
						<pre> +---w public\ </pre>
\-key?						
						<pre>   bina\ </pre>
\ry						
						<pre> +---w privat\ </pre>
\e-key?						
						<pre>   bina\ </pre>
\ry						
						<pre> +--:(keystore) {keystore-suppo\ </pre>
\rted}?						
						<pre> +--rw keystore-refere\ </pre>
\nce?						
						<pre> ks:asymmetric\ </pre>
\-key-ref						
						<pre> +--:(certificate) +--rw certificate {sshcmn:ssh-x509-certs\ </pre>
\}}?						
						<pre> +--rw (local-or-keystore) +--:(local)   {local-keys-sup\ </pre>

\ported}?					+---rw local-definition
					+---rw algorithm?
					asymmetric\
\-key-algorithm-ref					
					+---rw public-key?
					binary
					+---rw private-key?
					union
					+----x generate-hid\
\den-key					
					+----w input
					+----w algori\
\thm					
					asym\
\metric-key-algorithm-ref					
					+----x install-hidd\
\en-key					
					+----w input
					+----w algori\
\thm					
					asym\
\metric-key-algorithm-ref					
					+----w public\
\-key?					
					bina\
\ry					
					+----w privat\
\e-key?					
					bina\
\ry					
					+---rw cert?
					end-entity\
\-cert-cms					
					+----n certificate-\
\expiration					
					+--- expiration-\
\date					
					yang:da\
\te-and-time					
					+---:(keystore)
					{keystore-suppo\
\rted}?}					
					+---rw keystore-refere\
\nce?					
					ks:asymmetric\
\-key-certificate-ref					
					+---rw ssh-client-cert-auth



				{sshcmn:ssh-x509-certs}?
			++rw	pinned-ca-certs?
				ta:pinned-certificates-ref
				{ta:x509-certificates}?
			++rw	pinned-client-certs?
				ta:pinned-certificates-ref
				{ta:x509-certificates}?
			++rw	ssh-transport-params
\g}?				{ssh-server-transport-params-confi\
			++rw	host-key
				++rw host-key-alg* identityref
			++rw	key-exchange
				++rw key-exchange-alg* identityref
			++rw	encryption
				++rw encryption-alg* identityref
			++rw	mac
				++rw mac-alg* identityref
\s}?			++rw	ssh-keepalives {ssh-server-keepalive\
			++rw	max-wait? uint16
			++rw	max-attempts? uint8
			++:(tls)	{tls-call-home}?
			++rw	tls
			++rw	remote-address inet:host
\ber			++rw	remote-port? inet:port-num\
			++rw	local-address? inet:ip-addre\
\ss			++rw	local-port? inet:port-num\
\ber			++rw	tcp-keepalives {tcp-client-keepalive\
\s}?			++rw	idle-time? uint16
			++rw	max-probes? uint16
			++rw	probe-interval? uint16
			++rw	tls-server-identity
			++rw	(local-or-keystore)
			++:(local)	{local-keys-supported}?
			++rw	local-definition
			++rw	algorithm?
\hm-ref				asymmetric-key-algorit\
			++rw	public-key?
				binary
			++rw	private-key?
				union
			++x	generate-hidden-key

\algorithm-ref				+---w input +---w algorithm asymmetric-key-a\
\algorithm-ref				+---x install-hidden-key +---w input +---w algorithm asymmetric-key-a\
\ary				+---w public-key?   bin\
\ary				+---w private-key?   bin\
\cate-ref				+---rw cert?         end-entity-cert-cms +---n certificate-expiration +--- expiration-date yang:date-and-time +---:(keystore) {keystore-supported}? +---rw keystore-reference? ks:asymmetric-key-certifi\
\s)?				+---rw tls-client-auth +---rw pinned-ca-certs?         ta:pinned-certificates-ref {ta:x509-certificates}? +---rw pinned-client-certs?         ta:pinned-certificates-ref {ta:x509-certificates}? +---rw cert-maps +---rw cert-to-name* [id] +---rw id                   uint32 +---rw fingerprint         x509c2n:tls-fingerprint +---rw map-type           identityref +---rw name               string +---rw tls-hello-params         {tls-server-hello-params-config}? +---rw tls-versions         +---rw tls-version*   identityref +---rw cipher-suites         +---rw cipher-suite*   identityref +---rw tls-keepalives {tls-server-keepalive\
				+---rw max-wait?           uint16 +---rw max-attempts?     uint8
				+---rw connection-type +---rw (connection-type)

```

    +--:(persistent-connection)
    |   +--rw persistent!
    +--:(periodic-connection)
    |   +--rw periodic!
    |       +--rw period?          uint16
    |       +--rw anchor-time?     yang:date-and-time
    |       +--rw idle-timeout?    uint16
    +--rw reconnect-strategy
    |   +--rw start-with?          enumeration
    |   +--rw max-attempts?        uint8

```

#### 4.2. Example Usage

The following example illustrates configuring a NETCONF server to listen for NETCONF client connections using both the SSH and TLS transport protocols, as well as configuring call-home to two NETCONF clients, one using SSH and the other using TLS.

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

===== NOTE: '\n' line wrapping per BCP XX (RFC XXXX) =====

```

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

  <!-- endpoints to listen for NETCONF connections on -->
  <listen>
    <endpoint> <!-- listening for SSH connections -->
      <name>netconf/ssh</name>
      <ssh>
        <local-address>192.0.2.7</local-address>
        <ssh-server-identity>
          <host-key>
            <name>deployment-specific-certificate</name>
            <public-key>
              <local-definition>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
\nf-crypto-types">ct:rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
              </local-definition>
            </public-key>
          </host-key>
        </ssh-server-identity>
      <ssh-client-cert-auth>
        <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinne\

```

```

\d-ca-certs>
  <pinned-client-certs>explicitly-trusted-client-certs</pinn\
\ed-client-certs>
  </ssh-client-cert-auth>
</ssh>
</endpoint>
<endpoint> <!-- listening for TLS sessions -->
  <name>netconf/tls</name>
  <tls>
    <local-address>192.0.2.7</local-address>
    <tls-server-identity>
      <local-definition>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cr\
\ypto-types">ct:rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </tls-server-identity>
    <tls-client-auth>
      <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinne\
\d-ca-certs>
      <pinned-client-certs>explicitly-trusted-client-certs</pinn\
\ed-client-certs>
      <cert-maps>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
          <map-type>x509c2n:san-any</map-type>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <fingerprint>B3:4F:A1:8C:54</fingerprint>
          <map-type>x509c2n:specified</map-type>
          <name>scooby-doo</name>
        </cert-to-name>
      </cert-maps>
    </tls-client-auth>
  </tls>
</endpoint>
</listen>

<!-- calling home to SSH and TLS based NETCONF clients -->
<call-home>
  <netconf-client> <!-- SSH-based client -->
    <name>config-mgr</name>
    <endpoints>
      <endpoint>

```

```

        <name>east-data-center</name>
        <ssh>
          <remote-address>east.config-mgr.example.com</remote-addr\
\ess>
          <ssh-server-identity>
            <host-key>
              <name>deployment-specific-certificate</name>
              <public-key>
                <local-definition>
                  <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang\
\:ietf-crypto-types">ct:rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                </local-definition>
              </public-key>
            </host-key>
          </ssh-server-identity>
          <ssh-client-cert-auth>
            <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
\inned-ca-certs>
            <pinned-client-certs>explicitly-trusted-client-certs</\
\pinned-client-certs>
          </ssh-client-cert-auth>
        </ssh>
      </endpoint>
    </endpoint>
    <name>west-data-center</name>
    <ssh>
      <remote-address>west.config-mgr.example.com</remote-addr\
\ess>
      <ssh-server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <local-definition>
              <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang\
\:ietf-crypto-types">ct:rsa2048</algorithm>
              <private-key>base64encodedvalue==</private-key>
              <public-key>base64encodedvalue==</public-key>
            </local-definition>
          </public-key>
        </host-key>
      </ssh-server-identity>
      <ssh-client-cert-auth>
        <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
\inned-ca-certs>
        <pinned-client-certs>explicitly-trusted-client-certs</\
\pinned-client-certs>

```

```

        </ssh-client-cert-auth>
      </ssh>
    </endpoint>
  </endpoints>
  <connection-type>
    <periodic>
      <idle-timeout>300</idle-timeout>
      <period>60</period>
    </periodic>
  </connection-type>
  <reconnect-strategy>
    <start-with>last-connected</start-with>
    <max-attempts>3</max-attempts>
  </reconnect-strategy>
</netconf-client>
<netconf-client> <!-- TLS-based client -->
  <name>data-collector</name>
  <endpoints>
    <endpoint>
      <name>east-data-center</name>
      <tls>
        <remote-address>east.analytics.example.com</remote-addre\
\ss>
        <tls-server-identity>
          <local-definition>
            <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
\rf-crypto-types">ct:rsa2048</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
            <cert>base64encodedvalue==</cert>
          </local-definition>
        </tls-server-identity>
        <tls-client-auth>
          <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
\inned-ca-certs>
          <pinned-client-certs>explicitly-trusted-client-certs</\
\pinned-client-certs>
          <cert-maps>
            <cert-to-name>
              <id>1</id>
              <fingerprint>11:0A:05:11:00</fingerprint>
              <map-type>x509c2n:san-any</map-type>
            </cert-to-name>
            <cert-to-name>
              <id>2</id>
              <fingerprint>B3:4F:A1:8C:54</fingerprint>
              <map-type>x509c2n:specified</map-type>
              <name>scooby-doo</name>

```

```

        </cert-to-name>
      </cert-maps>
    </tls-client-auth>
    <tcp-keepalives>
      <idle-time>15</idle-time>
      <max-probes>3</max-probes>
      <probe-interval>30</probe-interval>
    </tcp-keepalives>
    <tls-keepalives>
      <max-wait>30</max-wait>
      <max-attempts>3</max-attempts>
    </tls-keepalives>
  </tls>
</endpoint>
<endpoint>
  <name>west-data-center</name>
  <tls>
    <remote-address>west.analytics.example.com</remote-addre\
\ss>
    <tls-server-identity>
      <local-definition>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
\f-crypto-types">ct:rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </tls-server-identity>
    <tls-client-auth>
      <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
\inned-ca-certs>
      <pinned-client-certs>explicitly-trusted-client-certs</\
\pinned-client-certs>
    <cert-maps>
      <cert-to-name>
        <id>1</id>
        <fingerprint>11:0A:05:11:00</fingerprint>
        <map-type>x509c2n:san-any</map-type>
      </cert-to-name>
      <cert-to-name>
        <id>2</id>
        <fingerprint>B3:4F:A1:8C:54</fingerprint>
        <map-type>x509c2n:specified</map-type>
        <name>scooby-doo</name>
      </cert-to-name>
    </cert-maps>
  </tls-client-auth>
</tcp-keepalives>

```

```
        <idle-time>15</idle-time>
        <max-probes>3</max-probes>
        <probe-interval>30</probe-interval>
    </tcp-keepalives>
    <tls-keepalives>
        <max-wait>30</max-wait>
        <max-attempts>3</max-attempts>
    </tls-keepalives>
</tls>
</endpoint>
</endpoints>
<connection-type>
    <persistent/>
</connection-type>
<reconnect-strategy>
    <start-with>first-listed</start-with>
    <max-attempts>3</max-attempts>
</reconnect-strategy>
</netconf-client>
</call-home>
</netconf-server>
```

#### 4.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7407], [RFC7589], [RFC8071], [I-D.kwatsen-netconf-tcp-client-server], [I-D.ietf-netconf-ssh-client-server], and [I-D.ietf-netconf-tls-client-server].

This YANG module imports YANG types from [RFC6991], and YANG groupings from [RFC7407], [I-D.ietf-netconf-ssh-client-server] and [I-D.ietf-netconf-ssh-client-server].

```
<CODE BEGINS> file "ietf-netconf-server@2019-03-09.yang"
module ietf-netconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix ncs;

  import ietf-yang-types {
    prefix yang;
    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 AAAA: YANG Groupings for TCP Clients and TCP Servers";
  }

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

  import ietf-ssh-server {
    prefix sshs;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC YYYY: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-server {
    prefix tlss;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:    <http://datatracker.ietf.org/wg/netconf/>
    WG List:    <mailto:netconf@ietf.org>
    Author:     Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:     Gary Wu <mailto:garywu@cisco.com>
    Author:     Juergen Schoenwaelder
                  <mailto:j.schoenwaelder@jacobs-university.de>";
  description
    "This module contains a collection of YANG definitions for
    configuring NETCONF servers.

    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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Client and Server Models";
}

// Features

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

feature ssh-listen {
  description
    "The 'ssh-listen' feature indicates that the NETCONF server
    supports opening a port to accept NETCONF over SSH
    client connections.";
  reference
    "RFC 6242:
    Using the NETCONF Protocol over Secure Shell (SSH)";
}

feature tls-listen {
  description
    "The 'tls-listen' feature indicates that the NETCONF server
    supports opening a port to accept NETCONF over TLS
    client connections.";
  reference
    "RFC 7589: Using the NETCONF Protocol over Transport
```

```
        Layer Security (TLS) with Mutual X.509
        Authentication";
    }

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

    feature ssh-call-home {
        description
            "The 'ssh-call-home' feature indicates that the NETCONF
            server supports initiating a NETCONF over SSH call
            home connection to NETCONF clients.";
        reference
            "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
    }

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

    // Groupings

    grouping netconf-server-grouping {
        description
            "Top-level grouping for NETCONF server configuration.";
        container listen {
            if-feature "listen";
            presence "Enables server to listen for TCP connections";
            description
                "Configures listen behavior";
            leaf idle-timeout {
                type uint16;
                units "seconds";
                default 3600; // one hour
                description
                    "Specifies the maximum number of seconds that a NETCONF
```

```
    session may remain idle. A NETCONF session will be
    dropped if it is idle for an interval longer than this
    number of seconds.  If set to zero, then the server
    will never drop a session because it is idle.  Sessions
    that have a notification subscription active are never
    dropped.";
}
list endpoint {
  key "name";
  min-elements 1;
  description
    "List of endpoints to listen for NETCONF connections.";
  leaf name {
    type string;
    description
      "An arbitrary name for the NETCONF listen endpoint.";
  }
  choice transport {
    mandatory true;
    description
      "Selects between available transports.";
    case ssh {
      if-feature "ssh-listen";
      container ssh {
        description
          "SSH-specific listening configuration for inbound
          connections.";
        uses tcps:tcp-server-grouping {
          refine "local-port" {
            default "830";
            description
              "The NETCONF server will listen on the IANA-
              assigned well-known port value for 'netconf-ssh'
              (830) if no value is specified.";
          }
        }
        uses sshs:ssh-server-grouping;
      }
    }
    case tls {
      if-feature "tls-listen";
      container tls {
        description
          "TLS-specific listening configuration for inbound
          connections.";
        uses tcps:tcp-server-grouping {
          refine "local-port" {
            default "6513";
          }
        }
      }
    }
  }
}
```

```

        description
            "The NETCONF server will listen on the IANA-
            assigned well-known port value for 'netconf-tls'
            (6513) if no value is specified.";
    }
}
uses tlss:tls-server-grouping {
    refine "tls-client-auth" {
        must 'pinned-ca-certs or pinned-client-certs';
        description
            "NETCONF/TLS servers MUST validate client
            certificates.";
    }
    augment "tls-client-auth" {
        description
            "Augments in the cert-to-name structure.";
        container cert-maps {
            uses x509c2n:cert-to-name;
            description
                "The cert-maps container is used by a TLS-
                based NETCONF server to map the NETCONF
                client's presented X.509 certificate to a
                NETCONF username. If no matching and valid
                cert-to-name list entry can be found, then
                the NETCONF server MUST close the connection,
                and MUST NOT accept NETCONF messages over
                it.";
            reference
                "RFC WWWF: NETCONF over TLS, Section 7";
        }
    }
}
}
}
}
}
}
}
}
container call-home {
    if-feature "call-home";
    presence "Enables server to initiate TCP connections";
    description "Configures call-home behavior";
    list netconf-client {
        key "name";
        min-elements 1;
        description
            "List of NETCONF clients the NETCONF server is to
            initiate call-home connections to in parallel.";
        leaf name {

```

```
    type string;
    description
      "An arbitrary name for the remote NETCONF client.";
  }
  container endpoints {
    description
      "Container for the list of endpoints.";
    list endpoint {
      key "name";
      min-elements 1;
      ordered-by user;
      description
        "A non-empty user-ordered list of endpoints for this
        NETCONF server to try to connect to in sequence.
        Defining more than one enables high-availability.";
      leaf name {
        type string;
        description
          "An arbitrary name for this endpoint.";
      }
    }
    choice transport {
      mandatory true;
      description
        "Selects between available transports.";
      case ssh {
        if-feature "ssh-call-home";
        container ssh {
          description
            "Specifies SSH-specific call-home transport
            configuration.";
          uses tcpc:tcp-client-grouping {
            refine "remote-port" {
              default "4334";
              description
                "The NETCONF server will attempt to connect
                to the IANA-assigned well-known port for
                'netconf-ch-tls' (4334) if no value is
                specified.";
            }
          }
          uses sshs:ssh-server-grouping;
        }
      }
      case tls {
        if-feature "tls-call-home";
        container tls {
          description
            "Specifies TLS-specific call-home transport
```

```

        configuration.";
    uses tcpc:tcp-client-grouping {
        refine "remote-port" {
            default "4335";
            description
                "The NETCONF server will attempt to connect
                to the IANA-assigned well-known port for
                'netconf-ch-tls' (4335) if no value is
                specified.";
        }
    }
    uses tlss:tls-server-grouping {
        refine "tls-client-auth" {
            must 'pinned-ca-certs or pinned-client-certs';
            description
                "NETCONF/TLS servers MUST validate client
                certificates.";
        }
        augment "tls-client-auth" {
            description
                "Augments in the cert-to-name structure.";
            container cert-maps {
                uses x509c2n:cert-to-name;
                description
                    "The cert-maps container is used by a
                    TLS-based NETCONF server to map the
                    NETCONF client's presented X.509
                    certificate to a NETCONF username.  If
                    no matching and valid cert-to-name list
                    entry can be found, then the NETCONF
                    server MUST close the connection, and
                    MUST NOT accept NETCONF messages over
                    it.";
                reference
                    "RFC 7748: NETCONF over TLS, Section 7";
            }
        }
    }
} // tls
} // choice
} // endpoint
} // endpoints
container connection-type {
    description
        "Indicates the NETCONF server's preference for how the
        NETCONF 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 NETCONF
        client. If the connection goes down, immediately
        start trying to reconnect to it, using the
        reconnection strategy.

        This connection type minimizes any NETCONF client
        to NETCONF server data-transfer delay, albeit at
        the expense of holding resources longer.";
  } // container persistent
} // case persistent-connection
case periodic-connection {
  container periodic {
    presence "Indicates that a periodic connection is
      to be maintained.";
    description
      "Periodically connect to the NETCONF client. The
        NETCONF client should close the underlying TLS
        connection upon completing planned activities.

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



```
        example, for an anchor time is 15 minutes past
        midnight and a period interval of 24 hours, then
        a periodic connection will occur 15 minutes past
        midnight everyday.";
    }
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default 120; // two minutes
        description
            "Specifies the maximum number of seconds that
            a NETCONF session may remain idle. A NETCONF
            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 periodic
} // case periodic-connection
} // choice connection-type
} // container connection-type
container reconnect-strategy {
    description
        "The reconnection strategy directs how a NETCONF server
        reconnects to a NETCONF client, after discovering its
        connection to the client has dropped, even if due to a
        reboot. The NETCONF server starts with the specified
        endpoint and tries to connect to it max-attempts times
        before trying the next endpoint in the list (round
        robin).";
    leaf start-with {
        type enumeration {
            enum first-listed {
                description
                    "Indicates that reconnections should start with
                    the first endpoint listed.";
            }
            enum last-connected {
                description
                    "Indicates that reconnections should start with
                    the endpoint last connected to. If no previous
                    connection has ever been established, then the
                    first endpoint configured is used. NETCONF
                    servers SHOULD be able to remember the last
                    endpoint connected to across reboots.";
            }
            enum random-selection {
                description
```

```
        "Indicates that reconnections should start with
        a random endpoint.";
    }
}
default "first-listed";
description
    "Specifies which of the NETCONF client's endpoints
    the NETCONF server should start with when trying
    to connect to the NETCONF client.";
}
leaf max-attempts {
    type uint8 {
        range "1..max";
    }
    default "3";
    description
        "Specifies the number times the NETCONF server tries
        to connect to a specific endpoint before moving on
        to the next endpoint in the list (round robin).";
}
} // container reconnect-strategy
} // list netconf-client
} // container call-home
} // grouping netconf-server-grouping

// Protocol accessible node, for servers that implement this
// module.

container netconf-server {
    uses netconf-server-grouping;
    description
        "Top-level container for NETCONF server configuration.";
}
}
<CODE ENDS>
```

## 5. Design Considerations

Editorial: this section is a hold over from before, previously called "Objectives". It was only written to support the "server" (not the "client"). The question is if it's better to add the missing "client" parts, or remove this section altogether.

The primary purpose of the YANG modules defined herein is to enable the configuration of the NETCONF client and servers. This scope includes the following objectives:

### 5.1. Support all NETCONF transports

The YANG module should support all current NETCONF transports, namely NETCONF over SSH [RFC6242], NETCONF over TLS [RFC7589], and to be extensible to support future transports as necessary.

Because implementations may not support all transports, the modules should use YANG "feature" statements so that implementations can accurately advertise which transports are supported.

### 5.2. Enable each transport to select which keys to use

Servers may have a multiplicity of host-keys or server-certificates from which subsets may be selected for specific uses. For instance, a NETCONF server may want to use one set of SSH host-keys when listening on port 830, and a different set of SSH host-keys when calling home. The data models provided herein should enable configuration of which keys to use on a per-use basis.

### 5.3. Support authenticating NETCONF clients certificates

When a certificate is used to authenticate a NETCONF client, there is a need to configure the server to know how to authenticate the certificates. The server should be able to authenticate the client's certificate either by using path-validation to a configured trust anchor or by matching the client-certificate to one previously configured.

### 5.4. Support mapping authenticated NETCONF client certificates to usernames

When a client certificate is used for TLS client authentication, the NETCONF server must be able to derive a username from the authenticated certificate. Thus the modules defined herein should enable this mapping to be configured.

### 5.5. Support both listening for connections and call home

The NETCONF protocols were originally defined as having the server opening a port to listen for client connections. More recently the NETCONF working group defined support for call-home ([RFC8071]), enabling the server to initiate the connection to the client. Thus the modules defined herein should enable configuration for both listening for connections and calling home. Because implementations may not support both listening for connections and calling home, YANG "feature" statements should be used so that implementation can accurately advertise the connection types it supports.

## 5.6. For Call Home connections

The following objectives only pertain to call home connections.

### 5.6.1. Support more than one NETCONF client

A NETCONF server may be managed by more than one NETCONF client. For instance, a deployment may have one client for provisioning and another for fault monitoring. Therefore, when it is desired for a server to initiate call home connections, it should be able to do so to more than one client.

### 5.6.2. Support NETCONF clients having more than one endpoint

A NETCONF client managing a NETCONF server may implement a high-availability strategy employing a multiplicity of active and/or passive endpoint. Therefore, when it is desired for a server to initiate call home connections, it should be able to connect to any of the client's endpoints.

### 5.6.3. Support a reconnection strategy

Assuming a NETCONF client has more than one endpoint, then it becomes necessary to configure how a NETCONF server should reconnect to the client should it lose its connection to one the client's endpoints. For instance, the NETCONF server may start with first endpoint defined in a user-ordered list of endpoints or with the last endpoints it was connected to.

### 5.6.4. Support both persistent and periodic connections

NETCONF clients may vary greatly on how frequently they need to interact with a NETCONF server, how responsive interactions need to be, and how many simultaneous connections they can support. Some clients may need a persistent connection to servers to optimize real-time interactions, while others prefer periodic interactions in order to minimize resource requirements. Therefore, when it is necessary for server to initiate connections, it should be configurable if the connection is persistent or periodic.

### 5.6.5. Reconnection strategy for periodic connections

The reconnection strategy should apply to both persistent and periodic connections. How it applies to periodic connections becomes clear when considering that a periodic "connection" is a logical connection to a single server. That is, the periods of unconnectedness are intentional as opposed to due to external reasons. A periodic "connection" should always reconnect to the same

server until it is no longer able to, at which time the reconnection strategy guides how to connect to another server.

#### 5.6.6. Keep-alives for persistent connections

If a persistent connection is desired, it is the responsibility of the connection initiator to actively test the "aliveness" of the connection. The connection initiator must immediately work to reestablish a persistent connection as soon as the connection is lost. How often the connection should be tested is driven by NETCONF client requirements, and therefore keep-alive settings should be configurable on a per-client basis.

#### 5.6.7. Customizations for periodic connections

If a periodic connection is desired, it is necessary for the NETCONF server to know how often it should connect. This frequency determines the maximum amount of time a NETCONF client may have to wait to send data to a server. A server may connect to a client before this interval expires if desired (e.g., to send data to a client).

### 6. Security Considerations

The YANG module defined in this document uses groupings defined in [I-D.ietf-netconf-ssh-client-server] and [I-D.ietf-netconf-tls-client-server]. Please see the Security Considerations section in those documents for concerns related those groupings.

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

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

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

NONE

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

NONE

## 7. IANA Considerations

### 7.1. 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-netconf-client  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-server  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

### 7.2. The YANG Module Names Registry

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

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

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

## 8. References

### 8.1. Normative References

- [I-D.ietf-netconf-keystore]  
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-08 (work in progress), March 2019.
- [I-D.ietf-netconf-ssh-client-server]  
Watsen, K., Wu, G., and L. Xia, "YANG Groupings for SSH Clients and SSH Servers", draft-ietf-netconf-ssh-client-server-08 (work in progress), October 2018.
- [I-D.ietf-netconf-tls-client-server]  
Watsen, K., Wu, G., and L. Xia, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-08 (work in progress), October 2018.
- [I-D.kwatsen-netconf-tcp-client-server]  
Watsen, K., "YANG Groupings for TCP Clients and TCP Servers", draft-kwatsen-netconf-tcp-client-server-00 (work in progress), March 2019.
- [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>>.
- [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>>.

- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [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>>.
- [RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <<https://www.rfc-editor.org/info/rfc7589>>.
- [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>>.
- [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>>.

## 8.2. Informative References

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



## Appendix A. Change Log

## A.1. 00 to 01

- o Renamed "keychain" to "keystore".

## A.2. 01 to 02

- o Added to ietf-netconf-client ability to connected to a cluster of endpoints, including a reconnection-strategy.
- o Added to ietf-netconf-client the ability to configure connection-type and also keep-alive strategy.
- o Updated both modules to accommodate new groupings in the ssh/tls drafts.

## A.3. 02 to 03

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

## A.4. 03 to 04

- o Added RFC 8174 to Requirements Language Section.
- o Replaced refine statement in ietf-netconf-client to add a mandatory true.
- o Added refine statement in ietf-netconf-server to add a must statement.
- o Now there are containers and groupings, for both the client and server models.

## A.5. 04 to 05

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

## A.6. 05 to 06

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

## A.7. 06 to 07

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

## A.8. 07 to 08

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

## Appendix B. 08 to 09

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

## Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Ramkumar Dhanapal, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, and Bert Wijnen.

Author's Address

Kent Watsen  
Watsen Networks

EMail: [kent+ietf@watsen.net](mailto:kent+ietf@watsen.net)

Netconf  
Internet-Draft  
Intended status: Standards Track  
Expires: September 2, 2019

B. Lengyel  
Ericsson  
A. Clemm  
Huawei USA  
March 1, 2019

YangPush Notification Capabilities  
draft-ietf-netconf-notification-capabilities-01

Abstract

This document proposes a YANG module that allows a YANG server to specify server capabilities related to "Subscription to YANG Datastores" (YangPush). It proposes to use YANG Instance Data to document this information already in implementation time.

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 September 2, 2019.

Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Terminology . . . . .	2
2. Introduction . . . . .	2
3. Notification Capability Model . . . . .	4
3.1. Tree Diagram . . . . .	5
3.2. YANG Module . . . . .	6
4. Security Considerations . . . . .	9
5. IANA Considerations . . . . .	9
5.1. The IETF XML Registry . . . . .	9
5.2. The YANG Module Names Registry . . . . .	9
6. Open Issues . . . . .	9
7. References . . . . .	10
7.1. Normative References . . . . .	10
7.2. Informative References . . . . .	10
Appendix A. Changes between revisions . . . . .	10
Authors' Addresses . . . . .	10

## 1. Terminology

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

The terms Yang-Push, On-change subscription and Periodic subscription are used as defined in [I-D.ietf-netconf-yang-push]

On-change Notification Capability: The capability of the YANG server to support On-change subscriptions.

Implementation-time information: Information about the YANG server's behavior that is made available during the implementation of the server, available from a source other than a running Yang server.

Runtime-information: Information about the YANG server's behavior that is available from the running YANG server via a protocol like NETCONF, RESTCONF or HTTPS.

## 2. Introduction

As defined in [I-D.ietf-netconf-yang-push] a YANG server may allow clients to subscribe to updates from a datastore and subsequently push such update notifications to the client. Notifications may be sent periodically or on-change (more or less immediately after each change).

A YANG server supporting YANG-Push has a number of capabilities that are determined during the implementation of the server. These include:

- o Supported dampening periods for on-change subscriptions
- o Supported (reporting) periods for periodic subscriptions
- o Maximum number of objects that can be sent in an update
- o The set of data nodes for which on-change notification is supported

Servers MAY have limitations in how many update notifications and how many datastore node updates they can send out in a certain time-period.

In some cases, a publisher supporting on-change notifications will not be able to push updates for some object types on-change. Reasons for this might be that the value of the datastore node changes frequently (e.g. in-octets counter), that small object changes are frequent and meaningless (e.g., a temperature gauge changing 0.1 degrees), or that the implementation is not capable of on-change notification for a particular object. In those cases, it will be important for client applications to have a way to identify for which objects on-change notifications are supported and for which ones not.

Faced with the reality that support for on-change notification does not mean that such notifications will be sent for any specific data node, client/management applications can not rely on the on-change functionality unless the client has some means to identify for which objects on-change notifications are supported. YANG models are meant to be used as an interface contract. Without identification of data nodes supporting on-change, this contract would only state the YANG server may (or may not) send on-change notifications for a data node specified in a YANG module.

This document proposes a YANG module that allows a client to discover YANG-Push related capabilities.

YANG-Push related capability information will be needed both in implementation-time and run-time.

Implementation time information is needed by Network Management System (NMS) implementers. During NMS implementation for any functionality that depends on the notifications the information about on change notification capability is needed. If the information is not available early in some document, but only as instance data from

the network node, the NMS implementation will be delayed, because it has to wait for the network node to be ready. Also assuming that all NMS implementers will have a correctly configured network node available to retrieve data from, is an expensive proposition. (An NMS may handle dozens of network node types.) Often a fully functional NMS is a requirement for introducing a new network node type into a network, so delaying the NMS effectively delays the availability of the network node as well.

Implementation time information is needed by system integrators. When introducing a network node type into their network, operators often need to integrate the node type into their own management system. The NMS may have management functions that depend on on-change notifications. The network operator needs to plan his management practices and NMS implementation before he even decides to buy the specific network node type. Moreover the decision to buy the node type sometimes depends on these management possibilities.

Run-time information is needed

- o for any "purely model driven" client, e.g. a NETCONF-browser. As long as it has a valid model to read the capability information, it does not care which data nodes send notification, it will just handle what is available.
- o in case the capability might change during run-time e.g. due to licensing, HW constraints etc.
- o to check that early, implementation time capability information about the capabilities is indeed what the server implements (is the supplied documentation correct?)

### 3. Notification Capability Model

It is a goal to provide YangPush notification capability information in a format that is

- o vendor independent (standard)
- o formal (no freeform English text please)
- o the same both in implementation-time and run-time

The YANG module `ietf-notification-capabilities` is defined to provide the information. It contains

a set of capabilities related to the amount of notifications the server can send out

a default on-change notification capability separately for config false and config true data nodes

an on-change-notification-capability list containing a potentially different true/false notification capability for a few data nodes in the schema tree. Unless a node is in this list with a specific capability value, it inherits its on-change-notification-capability from its parent in the data tree, or from the relevant default values. It is assumed that only a small number of nodes will be included in this list: special cases where the default behavior is not followed. For a detailed description of the usage of this list see the description in the YANG module.

The information SHALL be provided in two ways both following the ietf-notification-capabilities module:

- o It SHALL be provided by the implementer as YANG instance data file complying to the [I-D.lengyel-netmod-yang-instance-data]. The file SHALL be available already in implementation time retrievable in a way that does not depend on a live network node. E.g. download from product Website.
- o It SHALL be available via NETCONF or RESTCONF from the live YANG server during runtime.

### 3.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model.

```

module: ietf-notification-capabilities
+--ro yangpush-notification-capabilities
  +--ro minimum-dampening-period?          uint32
  +--ro (update-period)?
    | +--:(minimum-update-period)
    | | +--ro minimum-update-period?      uint32
    | | +--:(supported-update-period)
    | | +--ro supported-update-period*    uint32
  +--ro max-objects-per-update?            uint32
  +--ro notification-sent-for-config-default?  boolean
  +--ro notification-sent-for-state-default?  boolean
  +--ro on-change-notification-capability* [node-selector]
    +--ro node-selector                    nacm:node-instance-identifier
    +--ro on-change-notification-sent      boolean

```



### 3.2. YANG Module

```
<CODE BEGINS> file "ietf-notification-capabilities.yang"

module ietf-notification-capabilities {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-notification-capabilities";
  prefix inc;

  import ietf-netconf-acm { prefix nacm; }

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

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

    WG Chair: Kent Watsen
               <mailto:kwatsen@juniper.net>

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

    Editor:   Balazs Lengyel
               <mailto:balazs.lengyel@ericsson.com>";

  description "This module specifies YANG-Push related server
    capabilities. It contains
    - capabilities related to the amount of notifications the
      server can send out
    - default and schema node specific information specifying
      the set of data nodes for which the YANG server is capable
      of sending on-change notifications.

    On-change notification capability is marked as true or false.
    This marking is inherited from the parent down the data tree
    unless explicitly marked otherwise.

    On-change notifications SHALL be sent for a config=true
    data node if one of the following is true:
    - if it is a top level data-node and is not specified in the
      on-change-notification-capability list and the
      notification-sent-for-config-default is true; or
    - notifications are sent for its parent data node and it is
      not specified in the on-change-notification-capability list; or
```

- it is specified in the on-change-notification-capability list and has a on-change-notification-sent value true.

On-change notifications SHALL be sent for a config=false data node if one of the following is true:

- if it is a top level data-node or has a config=true parent data node and is not specified in the on-change-notification-capability list and the notification-sent-for-state-default is true; or
  - notifications are sent for its parent data node which is also config=false and it is not specified in the on-change-notification-capability list; or
  - it is specified in the on-change-notification-capability list and has an on-change-notification-sent value true or
- ";

```
revision 2019-02-28 {
  description "Initial version";
  reference
    "RFC XXX: YangPush Notification Capabilities";
}

container yangpush-notification-capabilities {
  config false;
  description "YANG-Push related server capabilities";

  leaf minimum-dampening-period {
    type uint32;
    units msec;
    description "The minimum dampening period supported for on-change
      subscriptions.";
  }

  choice update-period {
    description "Supported period values.";
    leaf minimum-update-period {
      type uint32;
      units centiseconds;
      description "Minimum update period supported for a
        periodic subscription.";
    }

    leaf-list supported-update-period {
      type uint32;
      units centiseconds;
      description "Specific supported update period values
        for a periodic subscription";
    }
  }
}
```

```
}

leaf max-objects-per-update {
  type uint32;
  description "Maximum number of objects that can be sent
    in an update";
}

leaf notification-sent-for-config-default {
  type boolean;
  default true;
  description "Specifies the default value for
    top level configuration data nodes for the
    on-change-notification-sent capability.";
}

leaf notification-sent-for-state-default {
  type boolean;
  default false;
  description "Specifies the default value
    top level state data nodes for the
    on-change-notification-sent capability.";
}

list on-change-notification-capability {
  key node-selector ;
  description "A list of data nodes that have the
    on-change-notification-capability specifically defined.

    Should be used when specific data nodes support
    on-change notification in a module/subtree that
    generally does not support it or when some data nodes
    do not support the notification in a module/subtree
    that generally supports on-change notifications.";

  leaf node-selector {
    type nacm:node-instance-identifier;
    description "Selects the data nodes for which
      on-change capability is specified.";
  }

  leaf on-change-notification-sent {
    type boolean;
    mandatory true;
    description "Specifies whether the YANG server will
      send on-change notifications for the selected
      data nodes.";
  }
}
```

```
    }  
  }  
}
```

<CODE ENDS>

#### 4. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF and RESTCONF. Both of these protocols have mandatory-to- implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

The data in this module is not security sensitive.

#### 5. IANA Considerations

##### 5.1. The IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-notification-capabilities  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

##### 5.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the the following registrations are requested:

name: ietf-notification-capabilities  
namespace: urn:ietf:params:xml:ns:yang:ietf-notification-capabilities  
prefix: inc  
reference: RFC XXXX

#### 6. Open Issues

Do we need separate defaults/individual lists for every datastore?  
Proposal: no, it would be an overkill.

Should type nacm:node-instance-identifier be moved to yang-types?  
It is useful for more then just nacm.

## 7. References

### 7.1. Normative References

- [I-D.ietf-netconf-yang-push]  
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-22 (work in progress), February 2019.
- [I-D.lengyel-netmod-yang-instance-data]  
Lengyel, B. and B. Claise, "YANG Based Instance Data Files Format", draft-lengyel-netmod-yang-instance-data-05 (work in progress), October 2018.
- [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>>.

### 7.2. Informative References

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

## Appendix A. Changes between revisions

v00 - v01

- o Add more capabilities: minimum period, supported period max-number of objects, min dampening period, dampening supported

Authors' Addresses

Balazs Lengyel  
Ericsson  
Magyar Tudosok korutja 11  
1117 Budapest  
Hungary  
  
Email: balazs.lengyel@ericsson.com

Alexander Clemm  
Huawei USA  
2330 Central Expressway  
Santa Clara, CA 95050  
USA  
  
Email: ludwig@clemm.org

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
March 9, 2019

RESTCONF Client and Server Models  
draft-ietf-netconf-restconf-client-server-10

Abstract

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

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o I-D.ietf-netconf-keystore
- o I-D.ietf-netconf-tcp-client-server
- o I-D.ietf-netconf-tls-client-server
- o I-D.ietf-netconf-http-client-server

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

- o "XXXX" --> the assigned RFC value for this draft
- o "AAAA" --> the assigned RFC value for I-D.ietf-netconf-tcp-client-server
- o "CCCC" --> the assigned RFC value for I-D.ietf-netconf-tls-client-server

- o "BBBB" --> the assigned RFC value for I-D.ietf-netconf-http-client-server

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

#### Status of This Memo

This Internet-Draft 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 September 10, 2019.

#### Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.



## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
2. The RESTCONF Client Model . . . . .	4
2.1. Tree Diagram . . . . .	4
2.2. Example Usage . . . . .	9
2.3. YANG Module . . . . .	12
3. The RESTCONF Server Model . . . . .	20
3.1. Tree Diagram . . . . .	21
3.2. Example Usage . . . . .	24
3.3. YANG Module . . . . .	27
4. Security Considerations . . . . .	36
5. IANA Considerations . . . . .	37
5.1. The IETF XML Registry . . . . .	37
5.2. The YANG Module Names Registry . . . . .	38
6. References . . . . .	38
6.1. Normative References . . . . .	38
6.2. Informative References . . . . .	39
Appendix A. Change Log . . . . .	41
A.1. 00 to 01 . . . . .	41
A.2. 01 to 02 . . . . .	41
A.3. 02 to 03 . . . . .	41
A.4. 03 to 04 . . . . .	41
A.5. 04 to 05 . . . . .	41
A.6. 05 to 06 . . . . .	42
A.7. 06 to 07 . . . . .	42
A.8. 07 to 08 . . . . .	42
A.9. 08 to 09 . . . . .	42
A.10. 09 to 10 . . . . .	42
Acknowledgements . . . . .	43
Author's Address . . . . .	43

## 1. Introduction

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

## 1.1. Terminology

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

## 2. The RESTCONF Client Model

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

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

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

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

### 2.1. Tree Diagram

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

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```

module: ietf-restconf-client
  +--rw restconf-client
    +--rw initiate! {initiate}?
      +--rw restconf-server* [name]
        +--rw name string
        +--rw endpoints
          +--rw endpoint* [name]
            +--rw name string
            +--rw (transport)
              +--:(https) {https-initiate}?
                +--rw https
                  +--rw remote-address inet:host
                  +--rw remote-port?
                    | inet:port-number
                  +--rw local-address? inet:ip-addr\
\ess
                +--rw local-port?
                  | inet:port-number
                +--rw tcp-keepalives {tcp-client-keepalive\
\s}?
                  +--rw idle-time? uint16
                  +--rw max-probes? uint16

```

					+++rw probe-interval? uint16
					+++rw tls-client-identity
					+++rw (auth-type)
					+++:(certificate)
					+++rw certificate
					+++rw (local-or-keystore)
					+++:(local)
					{local-keys-support\
\ted)?					
					+++rw local-definition
					+++rw algorithm?
					asymmetric-ke\
\y-algorithm-ref					
					+++rw public-key?
					binary
					+++rw private-key?
					union
					++++x generate-hidden\
\-key					
					+++w input
					+++w algorithm
					asymmet\
\ric-key-algorithm-ref					
					++++x install-hidden-\
\key					
					+++w input
					+++w algorithm
					asymmetric\
\ric-key-algorithm-ref					
					+++w public-ke\
\y?					
					binary
					+++w private-k\
\ey?					
					binary
					+++rw cert?
					end-entity-ce\
\rt-cms					
					++++n certificate-exp\
\iration					
					+++ expiration-date
					yang:date-\
\and-time					
					+++:(keystore)
					{keystore-supporte\
\d)?					
					+++rw keystore-reference?
					ks:asymmetric-ke\

\y-certificate-ref

```

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

```

\s}?

```

|   +--rw max-wait?        uint16
|   +--rw max-attempts?    uint8
+--rw http-client-identity
|   +--rw (auth-type)?
|       +--:(basic)
|           +--rw basic
|               +--rw user-id?    string
|               +--rw password?   string
|       +--:(bearer)
|           +--rw bearer
|               +--rw token?      string
|       +--:(digest)
|           +--rw digest
|               +--rw username?   string
|               +--rw password?   string
|       +--:(hoba)
|           +--rw hoba
|       +--:(mutual)
|           +--rw mutual
|       +--:(negotiate)
|           +--rw negotiate
|       +--:(oauth)
|           +--rw oauth
|       +--:(scram-sha-1)
|           +--rw scram-sha-1
|       +--:(scram-sha-256)
|           +--rw scram-sha-256
|       +--:(vapid)
|           +--rw vapid
+--rw http-keepalives
|   {http-client-keepalives}?
+--rw max-wait?        uint16

```

```

|                                     +--rw max-attempts?   uint8
+--rw connection-type
|   +--rw (connection-type)
|   |   +--:(persistent-connection)
|   |   |   +--rw persistent!
|   |   +--:(periodic-connection)
|   |   |   +--rw periodic!
|   |   |   +--rw period?           uint16
|   |   |   +--rw anchor-time?     yang:date-and-time
|   |   |   +--rw idle-timeout?    uint16
+--rw reconnect-strategy
|   +--rw start-with?               enumeration
|   +--rw max-attempts?            uint8
+--rw listen! {listen}?
|   +--rw idle-timeout?            uint16
|   +--rw endpoint* [name]
|   |   +--rw name                  string
|   +--rw (transport)
|   |   +--:(https) {https-listen}?
|   |   +--rw https
|   |   |   +--rw local-address      inet:ip-address
|   |   |   +--rw local-port?       inet:port-number
|   |   |   +--rw tcp-keepalives {tcp-server-keepalives}?
|   |   |   |   +--rw idle-time?     uint16
|   |   |   |   +--rw max-probes?    uint16
|   |   |   |   +--rw probe-interval? uint16
|   |   +--rw tls-client-identity
|   |   |   +--rw (auth-type)
|   |   |   |   +--:(certificate)
|   |   |   |   +--rw certificate
|   |   |   |   |   +--rw (local-or-keystore)
|   |   |   |   |   |   +--:(local) {local-keys-supported\
\}?
|   |   |   |   |   |   +--rw local-definition
|   |   |   |   |   |   |   +--rw algorithm?
|   |   |   |   |   |   |   |   asymmetric-key-algo\
\algorithm-ref
|   |   |   |   |   |   |   +--rw public-key?
|   |   |   |   |   |   |   |   binary
|   |   |   |   |   |   |   +--rw private-key?
|   |   |   |   |   |   |   |   union
|   |   |   |   |   |   |   |   +---x generate-hidden-key
|   |   |   |   |   |   |   |   |   +---w input
|   |   |   |   |   |   |   |   |   +---w algorithm
|   |   |   |   |   |   |   |   |   asymmetric-ke\
\y-algorithm-ref
|   |   |   |   |   |   |   |   |   +---x install-hidden-key
|   |   |   |   |   |   |   |   |   |   +---w input

```

```

\y-algorithm-ref
|
|
|
|      +---w algorithm
|      |      asymmetric-ke\
|
|
|      +---w public-key?
|      |      binary
|      +---w private-key?
|      |      binary
|      +---rw cert?
|      |      end-entity-cert-cms
|      +---n certificate-expiration
|      |      +--- expiration-date
|      |      |      yang:date-and-ti\
\me
|
|      +---:(keystore) {keystore-supporte\
\d}?
|
|      +---rw keystore-reference?
|      |      ks:asymmetric-key-cert\
\ificate-ref
+---rw tls-server-auth
|   +---rw pinned-ca-certs?
|   |   ta:pinned-certificates-ref
|   |   {ta:x509-certificates}?
|   +---rw pinned-server-certs?
|   |   ta:pinned-certificates-ref
|   |   {ta:x509-certificates}?
+---rw tls-hello-params
|   {tls-client-hello-params-config}?
+---rw tls-versions
|   +---rw tls-version*   identityref
+---rw cipher-suites
|   +---rw cipher-suite*   identityref
+---rw tls-keepalives {tls-client-keepalives}?
|   +---rw max-wait?       uint16
|   +---rw max-attempts?   uint8
+---rw http-client-identity
|   +---rw (auth-type)?
|   |   +---:(basic)
|   |   |   +---rw basic
|   |   |   |   +---rw user-id?   string
|   |   |   |   +---rw password?  string
|   |   +---:(bearer)
|   |   |   +---rw bearer
|   |   |   |   +---rw token?     string
|   |   +---:(digest)
|   |   |   +---rw digest
|   |   |   |   +---rw username?   string
|   |   |   |   +---rw password?   string
|   |   +---:(hoba)

```

```

|      |--rw hoba
|      |--:(mutual)
|      |--rw mutual
|      |--:(negotiate)
|      |--rw negotiate
|      |--:(oauth)
|      |--rw oauth
|      |--:(scram-sha-1)
|      |--rw scram-sha-1
|      |--:(scram-sha-256)
|      |--rw scram-sha-256
|      |--:(vapid)
|      |--rw vapid
|--rw http-keepalives {http-client-keepalives}?
    |--rw max-wait?      uint16
    |--rw max-attempts?  uint8

```

## 2.2. Example Usage

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

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

===== NOTE: '\!' line wrapping per BCP XX (RFC XXXX) =====

```

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

  <!-- RESTCONF servers to initiate connections to -->
  <initiate>
    <restconf-server>
      <name>corp-fw1</name>
      <endpoints>
        <endpoint>
          <name>corp-fw1.example.com</name>
          <https>
            <remote-address>corp-fw1.example.com</remote-address>
            <tcp-keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </tcp-keepalives>
            <tls-client-identity>
              <certificate>
                <local-definition>
                  <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\

```

```

\ietf-crypto-types">ct:rsa2048</algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <cert>base64encodedvalue==</cert>
  </local-definition>
</certificate>
</tls-client-identity>
<tls-server-auth>
  <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
\inned-ca-certs>
  <pinned-server-certs>explicitly-trusted-server-certs</\
\pinned-server-certs>
</tls-server-auth>
<tls-keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</tls-keepalives>
<http-client-identity>
  <basic>
    <user-id>bob</user-id>
    <password>secret</password>
  </basic>
</http-client-identity>
<http-keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</http-keepalives>
</https>
<connection-type>
  <persistent/>
</connection-type>
</endpoint>
<endpoint>
  <name>corp-fw2.example.com</name>
  <https>
    <remote-address>corp-fw2.example.com</remote-address>
    <tcp-keepalives>
      <idle-time>15</idle-time>
      <max-probes>3</max-probes>
      <probe-interval>30</probe-interval>
    </tcp-keepalives>
    <tls-client-identity>
      <certificate>
        <local-definition>
          <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\
\ietf-crypto-types">ct:rsa2048</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>

```



```

        <cert>base64encodedvalue==</cert>
      </local-definition>
    </certificate>
  </tls-client-identity>
  <tls-server-auth>
    <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
\inned-ca-certs>
    <pinned-server-certs>explicitly-trusted-server-certs</\
\pinned-server-certs>
  </tls-server-auth>
  <tls-keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </tls-keepalives>
  <http-client-identity>
    <basic>
      <user-id>bob</user-id>
      <password>secret</password>
    </basic>
  </http-client-identity>
  <http-keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </http-keepalives>
</https>
<connection-type>
  <persistent/>
</connection-type>
</endpoint>
</endpoints>
</restconf-server>
</initiate>

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

```

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

### 2.3. YANG Module

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

```
<CODE BEGINS> file "ietf-restconf-client@2019-03-09.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-client {
    prefix tcpc;
    reference
      "RFC AAAA: YANG Groupings for TCP Clients and TCP Servers";
  }

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

  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC BBBB: YANG Groupings for TLS Clients and TLS Servers";
  }
}
```

```
}

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

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

contact
  "WG Web:    <http://datatracker.ietf.org/wg/restconf/>
  WG List:    <mailto:netconf@ietf.org>
  Author:     Kent Watsen <mailto:kent+ietf@watsen.net>
  Author:     Gary Wu <mailto:garywu@cisco.com>";

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

  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.

  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: RESTCONF Client and Server Models";
}
```

```
// Features

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

feature https-initiate {
  if-feature "initiate";
  description
    "The 'https-initiate' feature indicates that the RESTCONF
    client supports initiating HTTPS connections to RESTCONF
    servers. This feature exists as HTTPS might not be a
    mandatory to implement transport in the future.";
  reference
    "RFC 8040: RESTCONF Protocol";
}

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

feature https-listen {
  if-feature "listen";
  description
    "The 'https-listen' feature indicates that the RESTCONF client
    supports opening a port to listen for incoming RESTCONF
    server call-home connections. This feature exists as not
    all RESTCONF clients may support RESTCONF call home.";
  reference
    "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

// Groupings

grouping restconf-client-grouping {
  description
    "Top-level grouping for RESTCONF client configuration.";
  container initiate {
    if-feature "initiate";
    presence "Enables client to initiate TCP connections";
    description
```

```
    "Configures client initiating underlying TCP connections.";
list restconf-server {
  key "name";
  min-elements 1;
  description
    "List of RESTCONF servers the RESTCONF client is to
    initiate connections to in parallel.";
  leaf name {
    type string;
    description
      "An arbitrary name for the RESTCONF server.";
  }
  container endpoints {
    description
      "Container for the list of endpoints.";
    list endpoint {
      key "name";
      min-elements 1;
      ordered-by user;
      description
        "A non-empty user-ordered list of endpoints for this
        RESTCONF client to try to connect to in sequence.
        Defining more than one enables high-availability.";
      leaf name {
        type string;
        description
          "An arbitrary name for this endpoint.";
      }
    }
    choice transport {
      mandatory true;
      description
        "Selects between available transports. This is a
        'choice' statement so as to support additional
        transport options to be augmented in.";
      case https {
        if-feature "https-initiate";
        container https {
          description
            "Specifies HTTPS-specific transport
            configuration.";
          uses tcpc:tcp-client-grouping {
            refine "remote-port" {
              default "443";
              description
                "The RESTCONF client will attempt to
                connect to the IANA-assigned well-known
                port value for 'https' (443) if no value
                is specified.";
```

```
    }
  }
  uses tlsc:tls-client-grouping {
    refine "tls-client-identity/auth-type" {
      mandatory true;
      description
        "RESTCONF clients MUST pass some
        authentication credentials.";
    }
  }
  uses httpc:http-client-grouping;
}
} // https
} // transport
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 it,
          using the reconnection strategy. This
          connection type minimizes any RESTCONF server
          to RESTCONF client data-transfer delay, albeit
          at the expense of holding resources longer.";
      }
    }
    case periodic-connection {
      container periodic {
        presence "Indicates that a periodic connection is
        to be maintained.";
        description
          "Periodically connect to the RESTCONF server.
          The RESTCONF server should close the
          underlying TCP connection upon completing
          planned activities.

          This connection type increases resource
          utilization, albeit with increased delay in
```

```

        RESTCONF server to RESTCONF client
        interactions.";
    leaf period {
        type uint16;
        units "minutes";
        default "60";
        description
            "Duration of time between periodic
            connections.";
    }
    leaf anchor-time {
        type yang:date-and-time {
            // constrained to minute-level granularity
            pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                + '(Z|[\+|-]\d{2}:\d{2})';
        }
        description
            "Designates a timestamp before or after which
            a series of periodic connections are
            determined. The periodic connections occur
            at a whole multiple interval from the anchor
            time. For example, for an anchor time is 15
            minutes past midnight and a period interval
            of 24 hours, then a periodic connection will
            occur 15 minutes past midnight everyday.";
    }
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default 120; // two minutes
        description
            "Specifies the maximum number of seconds
            that the underlying 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.";
    }
}
} // periodic-connection
} // connection-type
} // connection-type
container reconnect-strategy {
    description
        "The reconnection strategy directs how a RESTCONF
        client reconnects to a RESTCONF server, after
        discovering its connection to the server has

```

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



```
    } // endpoints
  } // restconf-server
} // initiate

container listen {
  if-feature "listen";
  presence "Enables client to accept call-home connections";
  description
    "Configures client accepting call-home TCP connections.";
  leaf idle-timeout {
    type uint16;
    units "seconds";
    default 3600; // one hour
    description
      "Specifies the maximum number of seconds that an
       underlying 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. Sessions that have a notification subscription
       active are never dropped.";
  }
  list endpoint {
    key "name";
    min-elements 1;
    description
      "List of endpoints to listen for RESTCONF connections.";
    leaf name {
      type string;
      description
        "An arbitrary name for the RESTCONF listen endpoint.";
    }
  }
  choice transport {
    mandatory true;
    description
      "Selects between available transports. This is a
       'choice' statement so as to support additional
       transport options to be augmented in.";
    case https {
      if-feature "https-listen";
      container https {
        description
          "HTTPS-specific listening configuration for inbound
           connections.";
        uses tcps:tcp-server-grouping {
          refine "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.";
    }
}
uses tlsc:tls-client-grouping {
    refine "tls-client-identity/auth-type" {
        mandatory true;
        description
            "RESTCONF clients MUST pass some authentication
            credentials.";
    }
}
uses httpc:http-client-grouping;
}
} // case https
} // transport
} // endpoint
} // listen
} // restconf-client

// Protocol accessible node, for servers that implement this
// module.

container restconf-client {
    uses restconf-client-grouping;
    description
        "Top-level container for RESTCONF client configuration.";
}
}
<CODE ENDS>
```

### 3. The RESTCONF Server Model

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

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

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

### 3.1. Tree Diagram

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

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```

module: ietf-restconf-server
  +--rw restconf-server
    +--rw listen! {listen}?
      +--rw endpoint* [name]
        +--rw name          string
        +--rw (transport)
          +--:(https) {https-listen}?
            +--rw https
              +--rw local-address          inet:ip-address
              +--rw local-port?            inet:port-number
              +--rw tcp-keepalives {tcp-server-keepalives}?
                +--rw idle-time?          uint16
                +--rw max-probes?         uint16
                +--rw probe-interval?     uint16
              +--rw tls-server-identity
                +--rw (local-or-keystore)
                  +--:(local) {local-keys-supported}?
                    +--rw local-definition
                      +--rw algorithm?
                        | asymmetric-key-algorithm-ref
                      +--rw public-key?    bina\
\ry
                        +--rw private-key?    union
                        +---x generate-hidden-key
                          +---w input
                            +---w algorithm
                              asymmetric-key-algorit\
\hm-ref
                        +---x install-hidden-key
                          +---w input
                            +---w algorithm
                              asymmetric-key-algorit\
\hm-ref
                          +---w public-key?    binary
                          +---w private-key?    binary
                        +--rw cert?
                          | end-entity-cert-cms
                        +---n certificate-expiration
                          +-- expiration-date

```

```

|                                     yang:date-and-time
|                                     +---:(keystore) {keystore-supported}?
|                                     +---rw keystore-reference?
|                                     ks:asymmetric-key-certificate-r\
\ef
|
| +---rw tls-client-auth
| | +---rw pinned-ca-certs?
| | | ta:pinned-certificates-ref
| | | {ta:x509-certificates}?
| | +---rw pinned-client-certs?
| | | ta:pinned-certificates-ref
| | | {ta:x509-certificates}?
| | +---rw cert-maps
| | | +---rw cert-to-name* [id]
| | | | +---rw id                               uint32
| | | | +---rw fingerprint
| | | | | x509c2n:tls-fingerprint
| | | | +---rw map-type                         identityref
| | | | +---rw name                             string
| | +---rw tls-hello-params
| | | {tls-server-hello-params-config}?
| | +---rw tls-versions
| | | +---rw tls-version*                       identityref
| | +---rw cipher-suites
| | | +---rw cipher-suite*                     identityref
| | +---rw tls-keepalives {tls-server-keepalives}?
| | | +---rw max-wait?                          uint16
| | | +---rw max-attempts?                      uint8
| | +---rw http-keepalives {http-server-keepalives}?
| | | +---rw max-wait?                          uint16
| | | +---rw max-attempts?                      uint8
+---rw call-home! {call-home}?
| +---rw restconf-client* [name]
| | +---rw name                                string
| | +---rw endpoints
| | | +---rw endpoint* [name]
| | | | +---rw name                            string
| | | | +---rw (transport)
| | | | +---:(https) {https-call-home}?
| | | | +---rw https
| | | | | +---rw remote-address                inet:host
| | | | | +---rw remote-port?                 inet:port-num\
\ber
| | | | +---rw local-address?                 inet:ip-addre\
\ss
| | | | +---rw local-port?                    inet:port-num\
\ber
| | | | +---rw tcp-keepalives {tcp-client-keepalive}

```

\s)?	<pre> +--rw idle-time?          uint16 +--rw max-probes?         uint16 +--rw probe-interval?    uint16 +--rw tls-server-identity +--rw (local-or-keystore)   +--:(local) {local-keys-supported}?     +--rw local-definition       +--rw algorithm?         asymmetric-key-algorit\ </pre>
\hm-ref	<pre> +--rw public-key?         binary +--rw private-key?         union +---x generate-hidden-key   +---w input     +---w algorithm       asymmetric-key-a\ </pre>
\lgorithm-ref	<pre> +---x install-hidden-key   +---w input     +---w algorithm       asymmetric-key-a\ </pre>
\lgorithm-ref	<pre> +---w public-key?    bin\ </pre>
\ary	<pre> +---w private-key?  bin\ </pre>
\ary	<pre> +--rw cert?         end-entity-cert-cms +---n certificate-expiration   +-- expiration-date     yang:date-and-time +--:(keystore) {keystore-supported}? +--rw keystore-reference?   ks:asymmetric-key-certifi\ </pre>
\cate-ref	<pre> +--rw tls-client-auth +--rw pinned-ca-certs?         ta:pinned-certificates-ref     {ta:x509-certificates}? +--rw pinned-client-certs?         ta:pinned-certificates-ref     {ta:x509-certificates}? +--rw cert-maps   +--rw cert-to-name* [id]     +--rw id          uint32 </pre>

```

|
|
|         +--rw fingerprint
|         |         x509c2n:tls-fingerprint
|         +--rw map-type      identityref
|         +--rw name          string
+--rw tls-hello-params
|   {tls-server-hello-params-config}?
|   +--rw tls-versions
|   |   +--rw tls-version*    identityref
|   +--rw cipher-suites
|   |   +--rw cipher-suite*   identityref
+--rw tls-keepalives {tls-server-keepalive\
\s}?
|
|   +--rw max-wait?          uint16
|   +--rw max-attempts?     uint8
+--rw http-keepalives
|   {http-server-keepalives}?
|   +--rw max-wait?          uint16
|   +--rw max-attempts?     uint8
+--rw connection-type
|   +--rw (connection-type)
|   |   +--:(persistent-connection)
|   |   |   +--rw persistent!
|   |   +--:(periodic-connection)
|   |   |   +--rw periodic!
|   |   |   +--rw period?          uint16
|   |   |   +--rw anchor-time?     yang:date-and-time
|   |   |   +--rw idle-timeout?    uint16
+--rw reconnect-strategy
|   +--rw start-with?          enumeration
|   +--rw max-attempts?       uint8

```

### 3.2. Example Usage

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

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

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```

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

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

```

```

    <endpoint>
      <name>netconf/tls</name>
      <https>
        <local-address>11.22.33.44</local-address>
        <tls-server-identity>
          <local-definition>
            <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cr\
\ypto-types">ct:rsa2048</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
            <cert>base64encodedvalue==</cert>
          </local-definition>
        </tls-server-identity>
        <tls-client-auth>
          <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinne\
\d-ca-certs>
          <pinned-client-certs>explicitly-trusted-client-certs</pinn\
\ed-client-certs>
          <cert-maps>
            <cert-to-name>
              <id>1</id>
              <fingerprint>11:0A:05:11:00</fingerprint>
              <map-type>x509c2n:san-any</map-type>
            </cert-to-name>
            <cert-to-name>
              <id>2</id>
              <fingerprint>B3:4F:A1:8C:54</fingerprint>
              <map-type>x509c2n:san-specified</map-type>
              <name>scooby-doo</name>
            </cert-to-name>
          </cert-maps>
        </tls-client-auth>
      </https>
    </endpoint>
  </listen>

  <!-- call home to a RESTCONF client with two endpoints -->
  <call-home>
    <restconf-client>
      <name>config-manager</name>
      <endpoints>
        <endpoint>
          <name>east-data-center</name>
          <https>
            <remote-address>east.example.com</remote-address>
            <tls-server-identity>
              <local-definition>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf\

```

```

\ f-crypto-types">ct:rsa2048</algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <cert>base64encodedvalue==</cert>
  </local-definition>
</tls-server-identity>
<tls-client-auth>
  <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
\inned-ca-certs>
  <pinned-client-certs>explicitly-trusted-client-certs</\
\pinned-client-certs>
  <cert-maps>
    <cert-to-name>
      <id>1</id>
      <fingerprint>11:0A:05:11:00</fingerprint>
      <map-type>x509c2n:san-any</map-type>
    </cert-to-name>
    <cert-to-name>
      <id>2</id>
      <fingerprint>B3:4F:A1:8C:54</fingerprint>
      <map-type>x509c2n:specified</map-type>
      <name>scooby-doo</name>
    </cert-to-name>
  </cert-maps>
</tls-client-auth>
</https>
</endpoint>
<endpoint>
  <name>west-data-center</name>
  <https>
    <remote-address>west.example.com</remote-address>
    <tcp-keepalives>
      <idle-time>15</idle-time>
      <max-probes>3</max-probes>
      <probe-interval>30</probe-interval>
    </tcp-keepalives>
    <tls-server-identity>
      <local-definition>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
\ f-crypto-types">ct:rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </tls-server-identity>
    <tls-client-auth>
      <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
\inned-ca-certs>

```



```

        <pinned-client-certs>explicitly-trusted-client-certs</\
\pinned-client-certs>
        <cert-maps>
            <cert-to-name>
                <id>1</id>
                <fingerprint>11:0A:05:11:00</fingerprint>
                <map-type>x509c2n:san-any</map-type>
            </cert-to-name>
            <cert-to-name>
                <id>2</id>
                <fingerprint>B3:4F:A1:8C:54</fingerprint>
                <map-type>x509c2n:specified</map-type>
                <name>scooby-doo</name>
            </cert-to-name>
        </cert-maps>
    </tls-client-auth>
    <http-keepalives>
        <max-wait>30</max-wait>
        <max-attempts>3</max-attempts>
    </http-keepalives>
</https>
</endpoint>
</endpoints>
<connection-type>
    <periodic>
        <idle-timeout>300</idle-timeout>
        <period>60</period>
    </periodic>
</connection-type>
<reconnect-strategy>
    <start-with>last-connected</start-with>
    <max-attempts>3</max-attempts>
</reconnect-strategy>
</restconf-client>
</call-home>
</restconf-server>

```

### 3.3. YANG Module

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

```

<CODE BEGINS> file "ietf-restconf-server@2019-03-09.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-x509-cert-to-name {
  prefix x509c2n;
  reference
    "RFC 7407: A YANG Data Model for SNMP Configuration";
}

import ietf-tcp-client {
  prefix tcpc;
  reference
    "RFC AAAA: YANG Groupings for TCP Clients and TCP Servers";
}

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

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

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

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

contact
  "WG Web:    <http://datatracker.ietf.org/wg/netconf/>
  WG List:    <mailto:netconf@ietf.org>
  Author:     Kent Watsen <mailto:kent+ietf@watsen.net>
  Author:     Gary Wu <mailto:garywu@cisco.com>
  Author:     Juergen Schoenwaelder
```

<mailto:j.schoenwaelder@jacobs-university.de>;

description

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

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

revision 2019-03-09 {

description

"Initial version";

reference

"RFC XXXX: RESTCONF Client and Server Models";

}

// Features

feature listen {

description

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

}

feature https-listen {

if-feature "listen";

description

"The 'https-listen' feature indicates that the RESTCONF server supports opening a port to listen for incoming RESTCONF client connections. This feature exists as HTTPS might not

```
        be a mandatory to implement transport in the future.";
    reference
        "RFC 8040: RESTCONF Protocol";
}

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

feature https-call-home {
    if-feature "call-home";
    description
        "The 'https-call-home' feature indicates that the RESTCONF
        server supports initiating connections to RESTCONF clients.
        This feature exists as not all RESTCONF servers may
        support RESTCONF call home.";
    reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

// Groupings

grouping restconf-server-grouping {
    description
        "Top-level grouping for RESTCONF server configuration.";
    container listen {
        if-feature "listen";
        presence "Enables server to listen for TCP connections";
        description "Configures listen behavior";
        list endpoint {
            key "name";
            min-elements 1;
            description
                "List of endpoints to listen for RESTCONF connections.";
            leaf name {
                type string;
                description
                    "An arbitrary name for the RESTCONF listen endpoint.";
            }
        }
        choice transport {
            mandatory true;
            description

```

```
"Selects between available transports. This is a
'choice' statement so as to support additional
transport options to be augmented in.";
case https {
  if-feature "https-listen";
  container https {
    description
      "HTTPS-specific listening configuration for inbound
      connections.";
    uses tcps:tcp-server-grouping {
      refine "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.";
      }
    }
    uses tlss:tls-server-grouping {
      refine "tls-client-auth" {
        must 'pinned-ca-certs or pinned-client-certs';
        description
          "RESTCONF servers MUST be able to validate
          clients.";
      }
      augment "tls-client-auth" {
        description
          "Augments in the cert-to-name structure,
          so the RESTCONF server can map TLS-layer
          client certificates to RESTCONF usernames.";
        container cert-maps {
          uses x509c2n:cert-to-name;
          description
            "The cert-maps container is used by a TLS-
            based RESTCONF server to map the RESTCONF
            client's presented X.509 certificate to
            a RESTCONF username. If no matching and
            valid cert-to-name list entry can be found,
            then the RESTCONF server MUST close the
            connection, and MUST NOT accept RESTCONF
            messages over it.";
          reference
            "RFC 7407: A YANG Data Model for SNMP
            Configuration.";
        }
      }
    }
  }
}
uses https:http-server-grouping;
```

```
        } // https container
    } // tls case
} // transport
} // endpoint
} // listen

container call-home {
    if-feature "call-home";
    presence "Enables server to initiate TCP connections";
    description "Configures call-home behavior";
    list restconf-client {
        key "name";
        min-elements 1;
        description
            "List of RESTCONF clients the RESTCONF server is to
            initiate call-home connections to in parallel.";
        leaf name {
            type string;
            description
                "An arbitrary name for the remote RESTCONF client.";
        }
    }
    container endpoints {
        description
            "Container for the list of endpoints.";
        list endpoint {
            key "name";
            min-elements 1;
            ordered-by user;
            description
                "User-ordered list of endpoints for this RESTCONF
                client. Defining more than one enables high-
                availability.";
            leaf name {
                type string;
                description
                    "An arbitrary name for this endpoint.";
            }
        }
        choice transport {
            mandatory true;
            description
                "Selects between available transports. This is a
                'choice' statement so as to support additional
                transport options to be augmented in.";
            case https {
                if-feature "https-call-home";
                container https {
                    description
                        "Specifies HTTPS-specific call-home transport
```

```
        configuration.";
    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.";
        }
    }
    uses tlss:tls-server-grouping {
        refine "tls-client-auth" {
            must 'pinned-ca-certs or pinned-client-certs';
            description
                "RESTCONF servers MUST be able to validate
                clients.";
        }
        augment "tls-client-auth" {
            description
                "Augments in the cert-to-name structure,
                so the RESTCONF server can map TLS-layer
                client certificates to RESTCONF usernames.";
            container cert-maps {
                uses x509c2n:cert-to-name;
                description
                    "The cert-maps container is used by a
                    TLS-based RESTCONF server to map the
                    RESTCONF client's presented X.509
                    certificate to a RESTCONF username. If
                    no matching and valid cert-to-name list
                    entry can be found, then the RESTCONF
                    server MUST close the connection, and
                    MUST NOT accept RESTCONF messages over
                    it.";
                reference
                    "RFC 7407: A YANG Data Model for SNMP
                    Configuration.";
            }
        }
    }
    uses https:http-server-grouping;
}

} // transport
} // endpoint
} // endpoints
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 it, using the
        reconnection strategy.

        This connection type minimizes any RESTCONF
        client to RESTCONF server data-transfer delay,
        albeit at the expense of holding resources
        longer.";
    }
  }
  case periodic-connection {
    container periodic {
      presence "Indicates that a periodic connection is
        to be maintained.";
      description
        "Periodically connect to the RESTCONF client. The
        RESTCONF client should close the underlying TCP
        connection upon completing planned activities.

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



```

        description
        "Designates a timestamp before or after which a
        series of periodic connections are determined.
        The periodic connections occur at a whole
        multiple interval from the anchor time. For
        example, for an anchor time is 15 minutes past
        midnight and a period interval of 24 hours, then
        a periodic connection will occur 15 minutes past
        midnight everyday.";
    }
    leaf idle-timeout {
        type uint16;
        units "seconds";
        default 120; // two minutes
        description
        "Specifies the maximum number of seconds that
        the underlying 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 no previous
                connection has ever been established, then the
                first endpoint configured is used. RESTCONF

```

```
        servers SHOULD be able to remember the last
        endpoint connected to across reboots.";
    }
    enum random-selection {
        description
            "Indicates that reconnections should start with
            a random endpoint.";
    }
}
default "first-listed";
description
    "Specifies which of the RESTCONF client's endpoints
    the RESTCONF server should start with when trying
    to connect to the RESTCONF client.";
}
leaf max-attempts {
    type uint8 {
        range "1..max";
    }
    default "3";
    description
        "Specifies the number times the RESTCONF server tries
        to connect to a specific endpoint before moving on to
        the next endpoint in the list (round robin).";
}
}
} // restconf-client
} // call-home
} // restconf-server-grouping

// Protocol accessible node, for servers that implement this
// module.

container restconf-server {
    uses restconf-server-grouping;
    description
        "Top-level container for RESTCONF server configuration.";
}
}
<CODE ENDS>
```

#### 4. Security Considerations

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

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

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

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

NONE

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

NONE

## 5. IANA Considerations

### 5.1. The IETF XML Registry

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-client  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-server  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

## 5.2. The YANG Module Names Registry

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

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

## 6. References

### 6.1. Normative References

[I-D.ietf-netconf-keystore]

Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-08 (work in progress), March 2019.

[I-D.ietf-netconf-tls-client-server]

Watsen, K., Wu, G., and L. Xia, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-08 (work in progress), October 2018.

[I-D.kwatsen-netconf-http-client-server]

Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", draft-kwatsen-netconf-http-client-server-00 (work in progress), March 2019.

[I-D.kwatsen-netconf-tcp-client-server]

Watsen, K., "YANG Groupings for TCP Clients and TCP Servers", draft-kwatsen-netconf-tcp-client-server-00 (work in progress), March 2019.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <<https://www.rfc-editor.org/info/rfc7407>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <<https://www.rfc-editor.org/info/rfc8071>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 6.2. Informative References

- [I-D.ietf-netconf-netconf-client-server]  
Watsen, K., "NETCONF Client and Server Models", draft-ietf-netconf-netconf-client-server-08 (work in progress), October 2018.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

## Appendix A. Change Log

## A.1. 00 to 01

- o Renamed "keychain" to "keystore".

## A.2. 01 to 02

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

## A.3. 02 to 03

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

## A.4. 03 to 04

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

## A.5. 04 to 05

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

## A.6. 05 to 06

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

## A.7. 06 to 07

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

## A.8. 07 to 08

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

## A.9. 08 to 09

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

## A.10. 09 to 10

- o Reformatted YANG modules.



## Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Ramkumar Dhanapal, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, and Bert Wijnen.

## Author's Address

Kent Watsen  
Watsen Networks

EMail: kent+ietf@watsen.net

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
G. Wu  
Cisco Systems  
L. Xia  
Huawei  
March 9, 2019

YANG Groupings for SSH Clients and SSH Servers  
draft-ietf-netconf-ssh-client-server-11

Abstract

This document defines three YANG modules: the first defines groupings for a generic SSH client, the second defines groupings for a generic SSH server, and the third defines common identities and groupings used by both the client and the server. It is intended that these groupings will be used by applications using the SSH protocol.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o I-D.ietf-netconf-trust-anchors
- o I-D.ietf-netconf-keystore

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

- o "XXXX" --> the assigned RFC value for this draft
- o "YYYY" --> the assigned RFC value for I-D.ietf-netconf-trust-anchors
- o "ZZZZ" --> the assigned RFC value for I-D.ietf-netconf-keystore

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

#### Status of This Memo

This Internet-Draft 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 September 10, 2019.

#### Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. The SSH Client Model . . . . .	4
3.1. Tree Diagram . . . . .	4
3.2. Example Usage . . . . .	5
3.3. YANG Module . . . . .	8
4. The SSH Server Model . . . . .	13
4.1. Tree Diagram . . . . .	13

4.2. Example Usage . . . . .	14
4.3. YANG Module . . . . .	18
5. The SSH Common Model . . . . .	23
5.1. Tree Diagram . . . . .	25
5.2. Example Usage . . . . .	26
5.3. YANG Module . . . . .	26
6. Security Considerations . . . . .	36
7. IANA Considerations . . . . .	37
7.1. The IETF XML Registry . . . . .	37
7.2. The YANG Module Names Registry . . . . .	38
8. References . . . . .	38
8.1. Normative References . . . . .	38
8.2. Informative References . . . . .	39
Appendix A. Change Log . . . . .	41
A.1. 00 to 01 . . . . .	41
A.2. 01 to 02 . . . . .	41
A.3. 02 to 03 . . . . .	41
A.4. 03 to 04 . . . . .	41
A.5. 04 to 05 . . . . .	42
A.6. 05 to 06 . . . . .	42
A.7. 06 to 07 . . . . .	42
A.8. 07 to 08 . . . . .	42
A.9. 08 to 09 . . . . .	42
A.10. 09 to 10 . . . . .	43
A.11. 10 to 11 . . . . .	43
Acknowledgements . . . . .	43
Authors' Addresses . . . . .	43

## 1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines a grouping for a generic SSH client, the second defines a grouping for a generic SSH server, and the third defines identities and groupings common to both the client and the server. It is intended that these groupings will be used by applications using the SSH protocol [RFC4252], [RFC4253], and [RFC4254]. For instance, these groupings could be used to help define the data model for an OpenSSH [OPENSSH] server or a NETCONF over SSH [RFC6242] based server.

The client and server YANG modules in this document each define one grouping, which is focused on just SSH-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen on or connect to. This affords applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "ssh-server-grouping"

grouping for the SSH parts it provides, while adding data nodes for the TCP-level call-home configuration.

The modules defined in this document use groupings defined in [I-D.ietf-netconf-keystore] enabling keys to be either locally defined or a reference to globally configured values.

The modules defined in this document optionally support [RFC6187] enabling X.509v3 certificate based host keys and public keys.

## 2. Terminology

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

## 3. The SSH Client Model

### 3.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-ssh-client" module that does not have groupings expanded.

```

module: ietf-ssh-client

grouping ssh-client-grouping
  +---u client-identity-grouping
  +---u server-auth-grouping
  +---u transport-params-grouping
  +---u keepalives-grouping
grouping client-identity-grouping
  +-- ssh-client-identity
    +-- username?          string
    +-- (auth-type)
      +--:(password)
        |  +-- password?    string
      +--:(public-key)
        |  +-- public-key
        |    +---u client-identity-grouping
      +--:(certificate)
        +-- certificate {sshcmn:ssh-x509-certs}?
        +---u client-identity-grouping
grouping server-auth-grouping
  +-- ssh-server-auth
    +-- pinned-ssh-host-keys?  ta:pinned-host-keys-ref
    |    {ta:ssh-host-keys}?
    +-- pinned-ca-certs?      ta:pinned-certificates-ref
    |    {sshcmn:ssh-x509-certs,ta:x509-certificates}?
    +-- pinned-server-certs?  ta:pinned-certificates-ref
    |    {sshcmn:ssh-x509-certs,ta:x509-certificates}?
grouping transport-params-grouping
  +-- ssh-transport-params {ssh-client-transport-params-config}?
  +---u transport-params-grouping
grouping keepalives-grouping
  +-- ssh-keepalives {ssh-client-keepalives}?
  +-- max-wait?      uint16
  +-- max-attempts?  uint8

```

### 3.2. Example Usage

This section presents two examples showing the `ssh-client-grouping` populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 3 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following example configures the client identity using a local key:

===== NOTE: '\!' line wrapping per BCP XX (RFC XXXX) =====

```
<ssh-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
  xmlns:alg="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- how this client will authenticate itself to the server -->
  <ssh-client-identity>
    <username>foobar</username>
    <public-key>
      <local-definition>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto\
\sh-types">ct:rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
      </local-definition>
    </public-key>
  </ssh-client-identity>

  <!-- which host-keys will this client trust -->
  <ssh-server-auth>
    <pinned-ssh-host-keys>explicitly-trusted-ssh-host-keys</pinned-s\
\sh-host-keys>
  </ssh-server-auth>

  <ssh-transport-params>
    <host-key>
      <host-key-alg>alg:ssh-rsa</host-key-alg>
    </host-key>
    <key-exchange>
      <key-exchange-alg>
        alg:diffie-hellman-group-exchange-sha256
      </key-exchange-alg>
    </key-exchange>
    <encryption>
      <encryption-alg>alg:aes256-ctr</encryption-alg>
      <encryption-alg>alg:aes192-ctr</encryption-alg>
      <encryption-alg>alg:aes128-ctr</encryption-alg>
      <encryption-alg>alg:aes256-cbc</encryption-alg>
      <encryption-alg>alg:aes192-cbc</encryption-alg>
      <encryption-alg>alg:aes128-cbc</encryption-alg>
    </encryption>
    <mac>
      <mac-alg>alg:hmac-sha2-256</mac-alg>
      <mac-alg>alg:hmac-sha2-512</mac-alg>
    </mac>
  </ssh-transport-params>
```

```
<ssh-keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</ssh-keepalives>

</ssh-client>
```

The following example configures the client identity using a key from the keystore:

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```
<ssh-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
  xmlns:alg="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- how this client will authenticate itself to the server -->
  <ssh-client-identity>
    <username>foobar</username>
    <public-key>
      <keystore-reference>ex-rsa-key</keystore-reference>
    </public-key>
  </ssh-client-identity>

  <!-- which host-keys will this client trust -->
  <ssh-server-auth>
    <pinned-ssh-host-keys>explicitly-trusted-ssh-host-keys</pinned-s\
\sh-host-keys>
  </ssh-server-auth>

  <ssh-transport-params>
    <host-key>
      <host-key-alg>alg:ssh-rsa</host-key-alg>
    </host-key>
    <key-exchange>
      <key-exchange-alg>
        alg:diffie-hellman-group-exchange-sha256
      </key-exchange-alg>
    </key-exchange>
    <encryption>
      <encryption-alg>alg:aes256-ctr</encryption-alg>
      <encryption-alg>alg:aes192-ctr</encryption-alg>
      <encryption-alg>alg:aes128-ctr</encryption-alg>
      <encryption-alg>alg:aes256-cbc</encryption-alg>
      <encryption-alg>alg:aes192-cbc</encryption-alg>
      <encryption-alg>alg:aes128-cbc</encryption-alg>
    </encryption>
    <mac>
```



```
        <mac-alg>algs:hmac-sha2-256</mac-alg>
        <mac-alg>algs:hmac-sha2-512</mac-alg>
    </mac>
</ssh-transport-params>

<ssh-keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
</ssh-keepalives>

</ssh-client>
```

### 3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors], and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-ssh-client@2019-03-09.yang"
module ietf-ssh-client {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
    prefix sshc;

    import ietf-ssh-common {
        prefix sshcmn;
        revision-date 2019-03-09; // stable grouping definitions
        reference
            "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
    }

    import ietf-trust-anchors {
        prefix ta;
        reference
            "RFC YYYY: YANG Data Model for Global Trust Anchors";
    }

    import ietf-keystore {
        prefix ks;
        reference
            "RFC ZZZZ:
            YANG Data Model for a Centralized Keystore Mechanism";
    }

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

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

WG List: <mailto:netconf@ietf.org>  
Author: Kent Watsen <mailto:kent+ietf@watsen.net>  
Author: Gary Wu <mailto:garywu@cisco.com>;

description

"This module defines reusable groupings for SSH clients that can be used as a basis for specific SSH client instances.

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";  
}  
  
// Features  
  
feature ssh-client-transport-params-config {  
  description  
    "SSH transport layer parameters are configurable on an SSH  
    client.";  
}  
  
feature ssh-client-keepalives {  
  description  
    "Per socket SSH keepalive parameters are configurable for  
    SSH clients on the server implementing this feature.";  
}
```

```
// Groupings

grouping ssh-client-grouping {
  description
    "A reusable grouping for configuring a SSH client without
    any consideration for how an underlying TCP session is
    established.";
  uses client-identity-grouping;
  uses server-auth-grouping;
  uses transport-params-grouping;
  uses keepalives-grouping;
}

grouping client-identity-grouping {
  description
    "A reusable grouping for configuring a SSH client identity.";
  container ssh-client-identity {
    description
      "The credentials used by the client to authenticate to
      the SSH server.";
    leaf username {
      type string;
      description
        "The username of this user. This will be the username
        used, for instance, to log into an SSH server.";
    }
    choice auth-type {
      mandatory true;
      description
        "The authentication type.";
      leaf password {
        type string;
        description
          "A password to be used for client authentication.";
      }
      container public-key {
        uses ks:local-or-keystore-asymmetric-key-grouping;
        description
          "A locally-defined or referenced asymmetric key pair
          to be used for client authentication.";
        reference
          "RFC ZZZZ:
          YANG Data Model for a Centralized Keystore Mechanism";
      }
      container certificate {
        if-feature "sshcmn:ssh-x509-certs";
        uses
          ks:local-or-keystore-end-entity-cert-with-key-grouping;
      }
    }
  }
}
```

```
        description
            "A locally-defined or referenced certificate
            to be used for client authentication.";
        reference
            "RFC ZZZZ
            YANG Data Model for a Centralized Keystore Mechanism";
    }
}
}
}

grouping server-auth-grouping {
    description
        "A reusable grouping for configuring SSH server
        authentication.";
    container ssh-server-auth {
        must 'pinned-ssh-host-keys or pinned-ca-certs or '
            + 'pinned-server-certs';
        description
            "Trusted server identities.";
        leaf pinned-ssh-host-keys {
            if-feature "ta:ssh-host-keys";
            type ta:pinned-host-keys-ref;
            description
                "A reference to a list of SSH host keys used by the
                SSH client to authenticate SSH server host keys.
                A server host key is authenticated if it is an exact
                match to a configured SSH host key.";
            reference
                "RFC YYYY: YANG Data Model for Global Trust Anchors";
        }
        leaf pinned-ca-certs {
            if-feature "sshcmn:ssh-x509-certs";
            if-feature "ta:x509-certificates";
            type ta:pinned-certificates-ref;
            description
                "A reference to a list of certificate authority (CA)
                certificates used by the SSH client to authenticate
                SSH server certificates. A server certificate is
                authenticated if it has a valid chain of trust to
                a configured CA certificate.";
            reference
                "RFC YYYY: YANG Data Model for Global Trust Anchors";
        }
        leaf pinned-server-certs {
            if-feature "sshcmn:ssh-x509-certs";
            if-feature "ta:x509-certificates";
```

```
    type ta:pinned-certificates-ref;
    description
      "A reference to a list of server certificates used by
      the SSH client to authenticate SSH server certificates.
      A server certificate is authenticated if it is an
      exact match to a configured server certificate.";
    reference
      "RFC YYYY: YANG Data Model for Global Trust Anchors";
  }
}

grouping transport-params-grouping {
  description
    "A reusable grouping for configuring a SSH transport
    parameters.";
  container ssh-transport-params {
    if-feature "ssh-client-transport-params-config";
    description
      "Configurable parameters of the SSH transport layer.";
    uses sshcmn:transport-params-grouping;
  }
}

grouping keepalives-grouping {
  description
    "A reusable grouping for configuring SSH client keepalive
    parameters.";
  container ssh-keepalives {
    if-feature "ssh-client-keepalives";
    description
      "Configures the keep-alive policy, to proactively test the
      aliveness of the SSH server. An unresponsive TLS server
      is dropped after approximately max-wait * max-attempts
      seconds.";
    leaf max-wait {
      type uint16 {
        range "1..max";
      }
      units "seconds";
      default "30";
      description
        "Sets the amount of time in seconds after which if no data
        has been received from the SSH server, a TLS-level message
        will be sent to test the aliveness of the SSH server.";
    }
    leaf max-attempts {
      type uint8;
    }
  }
}
```

```
        default "3";
        description
            "Sets the maximum number of sequential keep-alive messages
             that can fail to obtain a response from the SSH server
             before assuming the SSH server is no longer alive.";
    }
}
}
}
<CODE ENDS>
```

#### 4. The SSH Server Model

##### 4.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-ssh-server" module that does not have groupings expanded.

```

module: ietf-ssh-server

grouping ssh-server-grouping
  +---u server-identity-grouping
  +---u client-auth-grouping
  +---u transport-params-grouping
  +---u keepalives-grouping
grouping server-identity-grouping
  +-- ssh-server-identity
    +-- host-key* [name]
      +-- name? string
      +-- (host-key-type)
        +--:(public-key)
          | +-- public-key
          |   +---u server-identity-grouping
          +--:(certificate)
            +-- certificate {sshcmn:ssh-x509-certs}?
              +---u server-identity-grouping
grouping client-auth-grouping
  +-- ssh-client-cert-auth {sshcmn:ssh-x509-certs}?
  +-- pinned-ca-certs? ta:pinned-certificates-ref
    | {ta:x509-certificates}?
  +-- pinned-client-certs? ta:pinned-certificates-ref
    | {ta:x509-certificates}?
grouping transport-params-grouping
  +-- ssh-transport-params {ssh-server-transport-params-config}?
  +---u transport-params-grouping
grouping keepalives-grouping
  +-- ssh-keepalives {ssh-server-keepalives}?
  +-- max-wait? uint16
  +-- max-attempts? uint8

```

#### 4.2. Example Usage

This section presents two examples showing the `ssh-server-grouping` populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 3 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following example configures the server identity using a local key:

```

===== NOTE: '\\\' line wrapping per BCP XX (RFC XXXX) =====

<ssh-server xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"

```

```
xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

<!-- which host-keys will this SSH server present -->
<ssh-server-identity>
  <host-key>
    <name>deployment-specific-certificate</name>
    <public-key>
      <local-definition>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-cryp\
\to-types">ct:rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
      </local-definition>
    </public-key>
  </host-key>
</ssh-server-identity>

<!-- which client-certs will this SSH server trust -->
<ssh-client-cert-auth>
  <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
\erts>
  <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
\ent-certs>
</ssh-client-cert-auth>

<ssh-transport-params>
  <host-key>
    <host-key-alg>algs:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      algs:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>algs:aes256-ctr</encryption-alg>
    <encryption-alg>algs:aes192-ctr</encryption-alg>
    <encryption-alg>algs:aes128-ctr</encryption-alg>
    <encryption-alg>algs:aes256-cbc</encryption-alg>
    <encryption-alg>algs:aes192-cbc</encryption-alg>
    <encryption-alg>algs:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>algs:hmac-sha2-256</mac-alg>
    <mac-alg>algs:hmac-sha2-512</mac-alg>
  </mac>
</ssh-transport-params>
```



</ssh-server>

The following example configures the server identity using a key from the keystore:

===== NOTE: '\\\ ' line wrapping per BCP XX (RFC XXXX) =====

```
<ssh-server xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
            xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <!-- which host-keys will this SSH server present -->
  <ssh-server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <keystore-reference>ex-rsa-key</keystore-reference>
      </public-key>
    </host-key>
  </ssh-server-identity>

  <!-- which client-certs will this SSH server trust -->
  <ssh-client-cert-auth>
    <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
\erts>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
\ent-certs>
  </ssh-client-cert-auth>

  <ssh-transport-params>
    <host-key>
      <host-key-alg>algs:ssh-rsa</host-key-alg>
    </host-key>
    <key-exchange>
      <key-exchange-alg>
        algs:diffie-hellman-group-exchange-sha256
      </key-exchange-alg>
    </key-exchange>
    <encryption>
      <encryption-alg>algs:aes256-ctr</encryption-alg>
      <encryption-alg>algs:aes192-ctr</encryption-alg>
      <encryption-alg>algs:aes128-ctr</encryption-alg>
      <encryption-alg>algs:aes256-cbc</encryption-alg>
      <encryption-alg>algs:aes192-cbc</encryption-alg>
      <encryption-alg>algs:aes128-cbc</encryption-alg>
    </encryption>
    <mac>
      <mac-alg>algs:hmac-sha2-256</mac-alg>
      <mac-alg>algs:hmac-sha2-512</mac-alg>
    </mac>
  </ssh-transport-params>

</ssh-server>
```

#### 4.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore] and informative references to [RFC4253] and [RFC7317].

```
<CODE BEGINS> file "ietf-ssh-server@2019-03-09.yang"
module ietf-ssh-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
  prefix sshs;

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-trust-anchors {
    prefix ta;
    reference
      "RFC YYYY: YANG Data Model for Global Trust Anchors";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC ZZZZ:
        YANG Data Model for a Centralized Keystore Mechanism";
  }

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

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:   Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module defines reusable groupings for SSH servers that
    can be used as a basis for specific SSH server instances.

    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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
}

// Features

feature ssh-server-transport-params-config {
  description
    "SSH transport layer parameters are configurable on an SSH
    server.";
}

feature ssh-server-keepalives {
  description
    "Per socket SSH keepalive parameters are configurable for
    SSH servers on the server implementing this feature.";
}

// Groupings

grouping ssh-server-grouping {
  description
    "A reusable grouping for configuring a SSH server without
    any consideration for how underlying TCP sessions are
    established.";
  uses server-identity-grouping;
  uses client-auth-grouping;
  uses transport-params-grouping;
```

```
    uses keepalives-grouping;
}

grouping server-identity-grouping {
  description
    "A reusable grouping for configuring an SSH server identity.";
  container ssh-server-identity {
    description
      "The list of host-keys the SSH server will present when
      establishing a SSH connection.";
    list host-key {
      key "name";
      min-elements 1;
      ordered-by user;
      description
        "An ordered list of host keys the SSH server will use to
        construct its ordered list of algorithms, when sending
        its SSH_MSG_KEXINIT message, as defined in Section 7.1
        of RFC 4253.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer
        Protocol";
      leaf name {
        type string;
        description
          "An arbitrary name for this host-key";
      }
      choice host-key-type {
        mandatory true;
        description
          "The type of host key being specified";
        container public-key {
          uses ks:local-or-keystore-asymmetric-key-grouping;
          description
            "A locally-defined or referenced asymmetric key pair
            to be used for the SSH server's host key.";
          reference
            "RFC ZZZZ: YANG Data Model for a Centralized
            Keystore Mechanism";
        }
        container certificate {
          if-feature "sshcmn:ssh-x509-certs";
          uses
            ks:local-or-keystore-end-entity-cert-with-key-grouping;
          description
            "A locally-defined or referenced end-entity
            certificate to be used for the SSH server's
            host key.";
        }
      }
    }
  }
}
```

```
        reference
          "RFC ZZZZ: YANG Data Model for a Centralized
            Keystore Mechanism";
      }
    }
  }
}

grouping client-auth-grouping {
  description
    "A reusable grouping for configuring a SSH client
    authentication.";
  container ssh-client-cert-auth {
    if-feature "sshcmn:ssh-x509-certs";
    description
      "A reference to a list of pinned certificate authority (CA)
      certificates and a reference to a list of pinned client
      certificates.

      Note: password and public-key based client authentication
      are not configured in this YANG module as they are
      expected to be configured by the ietf-system module
      defined in RFC 7317.";
    reference
      "RFC 7317: A YANG Data Model for System Management";
    leaf pinned-ca-certs {
      if-feature "ta:x509-certificates";
      type ta:pinned-certificates-ref;
      description
        "A reference to a list of certificate authority (CA)
        certificates used by the SSH server to authenticate
        SSH client certificates. A client certificate is
        authenticated if it has a valid chain of trust to
        a configured pinned CA certificate.";
      reference
        "RFC YYYY: YANG Data Model for Global Trust Anchors";
    }
    leaf pinned-client-certs {
      if-feature "ta:x509-certificates";
      type ta:pinned-certificates-ref;
      description
        "A reference to a list of client certificates used by
        the SSH server to authenticate SSH client certificates.
        A clients certificate is authenticated if it is an
        exact match to a configured pinned client certificate.";
      reference
        "RFC YYYY: YANG Data Model for Global Trust Anchors";
    }
  }
}
```

```
    }
  }
}

grouping transport-params-grouping {
  description
    "A reusable grouping for configuring a SSH transport
    parameters.";
  container ssh-transport-params {
    if-feature "ssh-server-transport-params-config";
    description
      "Configurable parameters of the SSH transport layer.";
    uses sshcmn:transport-params-grouping;
  }
}

grouping keepalives-grouping {
  description
    "A reusable grouping for configuring SSH server keepalive
    parameters.";
  container ssh-keepalives {
    if-feature "ssh-server-keepalives";
    description
      "Configures the keep-alive policy, to proactively test the
      aliveness of the SSL client. An unresponsive SSL client
      is dropped after approximately max-wait * max-attempts
      seconds.";
    leaf max-wait {
      type uint16 {
        range "1..max";
      }
      units "seconds";
      default "30";
      description
        "Sets the amount of time in seconds after which if no data
        has been received from the SSL client, a SSL-level message
        will be sent to test the aliveness of the SSL client.";
    }
    leaf max-attempts {
      type uint8;
      default "3";
      description
        "Sets the maximum number of sequential keep-alive messages
        that can fail to obtain a response from the SSL client
        before assuming the SSL client is no longer alive.";
    }
  }
}
```

```
}  
<CODE ENDS>
```

## 5. The SSH Common Model

The SSH common model presented in this section contains identities and groupings common to both SSH clients and SSH servers. The transport-params-grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the SSH transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

[I-D.ietf-netconf-crypto-types] defines six categories of cryptographic algorithms (hash-algorithm, symmetric-key-encryption-algorithm, mac-algorithm, asymmetric-key-encryption-algorithm, signature-algorithm, key-negotiation-algorithm) and lists several widely accepted algorithms for each of them. The SSH client and server models use one or more of these algorithms. The SSH common model includes four parameters for configuring its permitted SSH algorithms, which are: host-key-alg, key-exchange-alg, encryption-alg and mac-alg. The following tables are provided, in part, to define the subset of algorithms defined in the crypto-types model used by SSH and, in part, to ensure compatibility of configured SSH cryptographic parameters for configuring its permitted SSH algorithms ("sshcmn" representing SSH common model, and "ct" representing crypto-types model which the SSH client/server model is based on):



sshcmn:host-key-alg	ct:signature-algorithm
dsa-sha1	dsa-sha1
rsa-pkcs1-sha1	rsa-pkcs1-sha1
rsa-pkcs1-sha256	rsa-pkcs1-sha256
rsa-pkcs1-sha512	rsa-pkcs1-sha512
ecdsa-secp256r1-sha256	ecdsa-secp256r1-sha256
ecdsa-secp384r1-sha384	ecdsa-secp384r1-sha384
ecdsa-secp521r1-sha512	ecdsa-secp521r1-sha512
x509v3-rsa-pkcs1-sha1	x509v3-rsa-pkcs1-sha1
x509v3-rsa2048-pkcs1-sha256	x509v3-rsa2048-pkcs1-sha1
x509v3-ecdsa-secp256r1-sha256	x509v3-ecdsa-secp256r1-sha256
x509v3-ecdsa-secp384r1-sha384	x509v3-ecdsa-secp384r1-sha384
x509v3-ecdsa-secp521r1-sha512	x509v3-ecdsa-secp521r1-sha512

Table 1 The SSH Host-key-alg Compatibility Matrix

sshcmn:key-exchange-alg	ct:key-negotiation-algorithm
diffie-hellman-group14-sha1	diffie-hellman-group14-sha1
diffie-hellman-group14-sha256	diffie-hellman-group14-sha256
diffie-hellman-group15-sha512	diffie-hellman-group15-sha512
diffie-hellman-group16-sha512	diffie-hellman-group16-sha512
diffie-hellman-group17-sha512	diffie-hellman-group17-sha512
diffie-hellman-group18-sha512	diffie-hellman-group18-sha512
ecdh-sha2-secp256r1	ecdh-sha2-secp256r1
ecdh-sha2-secp384r1	ecdh-sha2-secp384r1

Table 2 The SSH Key-exchange-alg Compatibility Matrix

sshcmn:encryption-alg	ct:symmetric-key-encryption-algorithm
aes-128-cbc	aes-128-cbc
aes-192-cbc	aes-192-cbc
aes-256-cbc	aes-256-cbc
aes-128-ctr	aes-128-ctr
aes-192-ctr	aes-192-ctr
aes-256-ctr	aes-256-ctr

Table 3 The SSH Encryption-alg Compatibility Matrix

sshcmn:mac-alg	ct:mac-algorithm
hmac-sha1	hmac-sha1
hmac-sha1-96	hmac-sha1-96
hmac-sha2-256	hmac-sha2-256
hmac-sha2-512	hmac-sha2-512

Table 4 The SSH Mac-alg Compatibility Matrix

As is seen in the tables above, the names of the "sshcmn" algorithms are all identical to the names of algorithms defined in [I-D.ietf-netconf-crypto-types]. While appearing to be redundant, it is important to realize that not all the algorithms defined in [I-D.ietf-netconf-crypto-types] are supported by SSH. That is, the algorithms supported by SSH are a subset of the algorithms defined in [I-D.ietf-netconf-crypto-types]. The algorithms used by SSH are redefined in this document in order to constrain the algorithms that may be selected to just the ones used by SSH.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive. As well, some algorithms that are REQUIRED by [RFC4253] are missing, notably "ssh-dss" and "diffie-hellman-group1-sha1" due to their weak security and there being alternatives that are widely supported.

### 5.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-ssh-common" module.

```
module: ietf-ssh-common
```

```

grouping transport-params-grouping
  +-- host-key
  |   +-- host-key-alg*   identityref
  +-- key-exchange
  |   +-- key-exchange-alg*   identityref
  +-- encryption
  |   +-- encryption-alg*   identityref
  +-- mac
      +-- mac-alg*   identityref

```

## 5.2. Example Usage

This following example illustrates how the transport-params-grouping appears when populated with some data.

```
<transport-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
  <host-key>
    <host-key-alg>algs:x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>algs:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      algs:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>algs:aes256-ctr</encryption-alg>
    <encryption-alg>algs:aes192-ctr</encryption-alg>
    <encryption-alg>algs:aes128-ctr</encryption-alg>
    <encryption-alg>algs:aes256-cbc</encryption-alg>
    <encryption-alg>algs:aes192-cbc</encryption-alg>
    <encryption-alg>algs:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>algs:hmac-sha2-256</mac-alg>
    <mac-alg>algs:hmac-sha2-512</mac-alg>
  </mac>
</transport-params>
```

## 5.3. YANG Module

This YANG module has normative references to [RFC4253], [RFC4344], [RFC4419], [RFC5656], [RFC6187], and [RFC6668].

```
<CODE BEGINS> file "ietf-ssh-common@2019-03-09.yang"
module ietf-ssh-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix sshcmn;

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

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

Author: Kent Watsen <mailto:kent+ietf@watsen.net>  
Author: Gary Wu <mailto:garywu@cisco.com>;

description

"This module defines a common features, identities, and groupings for Secure Shell (SSH)."

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";  
}
```

```
// Features
```

```
feature ssh-ecc {  
  description  
    "Elliptic Curve Cryptography is supported for SSH.";  
  reference  
    "RFC 5656: Elliptic Curve Algorithm Integration in the  
      Secure Shell Transport Layer";  
}
```

```
feature ssh-x509-certs {  
  description  
    "X.509v3 certificates are supported for SSH per RFC 6187.";  
  reference
```

```
        "RFC 6187: X.509v3 Certificates for Secure Shell
          Authentication";
    }

    feature ssh-dh-group-exchange {
        description
            "Diffie-Hellman Group Exchange is supported for SSH.";
        reference
            "RFC 4419: Diffie-Hellman Group Exchange for the
              Secure Shell (SSH) Transport Layer Protocol";
    }

    feature ssh-ctr {
        description
            "SDCTR encryption mode is supported for SSH.";
        reference
            "RFC 4344: The Secure Shell (SSH) Transport Layer
              Encryption Modes";
    }

    feature ssh-sha2 {
        description
            "The SHA2 family of cryptographic hash functions is
              supported for SSH.";
        reference
            "FIPS PUB 180-4: Secure Hash Standard (SHS)";
    }

    // Identities

    identity public-key-alg-base {
        description
            "Base identity used to identify public key algorithms.";
    }

    identity ssh-dss {
        base public-key-alg-base;
        description
            "Digital Signature Algorithm using SHA-1 as the
              hashing algorithm.";
        reference
            "RFC 4253:
              The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity ssh-rsa {
        base public-key-alg-base;
        description
```

```
        "RSASSA-PKCS1-v1_5 signature scheme using SHA-1 as the
        hashing algorithm.";
    reference
        "RFC 4253:
        The Secure Shell (SSH) Transport Layer Protocol";
}

identity ecdsa-sha2-nistp256 {
    base public-key-alg-base;
    if-feature "ssh-ecc and ssh-sha2";
    description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
        nistp256 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
}

identity ecdsa-sha2-nistp384 {
    base public-key-alg-base;
    if-feature "ssh-ecc and ssh-sha2";
    description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
        nistp384 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
}

identity ecdsa-sha2-nistp521 {
    base public-key-alg-base;
    if-feature "ssh-ecc and ssh-sha2";
    description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
        nistp521 curve and the SHA2 family of hashing algorithms.";
    reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
}

identity x509v3-ssh-rsa {
    base public-key-alg-base;
    if-feature "ssh-x509-certs";
    description
        "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
        in an X.509v3 certificate and using SHA-1 as the hashing
        algorithm.";
    reference
```

```
        "RFC 6187: X.509v3 Certificates for Secure Shell
          Authentication";
    }

    identity x509v3-rsa2048-sha256 {
        base public-key-alg-base;
        if-feature "ssh-x509-certs and ssh-sha2";
        description
            "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
             in an X.509v3 certificate and using SHA-256 as the hashing
             algorithm. RSA keys conveyed using this format MUST have a
             modulus of at least 2048 bits.";
        reference
            "RFC 6187: X.509v3 Certificates for Secure Shell
             Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp256 {
        base public-key-alg-base;
        if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
        description
            "Elliptic Curve Digital Signature Algorithm (ECDSA)
             using the nistp256 curve with a public key stored in
             an X.509v3 certificate and using the SHA2 family of
             hashing algorithms.";
        reference
            "RFC 6187: X.509v3 Certificates for Secure Shell
             Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp384 {
        base public-key-alg-base;
        if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
        description
            "Elliptic Curve Digital Signature Algorithm (ECDSA)
             using the nistp384 curve with a public key stored in
             an X.509v3 certificate and using the SHA2 family of
             hashing algorithms.";
        reference
            "RFC 6187: X.509v3 Certificates for Secure Shell
             Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp521 {
        base public-key-alg-base;
        if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
        description
            "Elliptic Curve Digital Signature Algorithm (ECDSA)
```

```
        using the nistp521 curve with a public key stored in
        an X.509v3 certificate and using the SHA2 family of
        hashing algorithms.";
    reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
        Authentication";
}

identity key-exchange-alg-base {
    description
        "Base identity used to identify key exchange algorithms.";
}

identity diffie-hellman-group14-sha1 {
    base key-exchange-alg-base;
    description
        "Diffie-Hellman key exchange with SHA-1 as HASH and
        Oakley Group 14 (2048-bit MODP Group).";
    reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group-exchange-sha1 {
    base key-exchange-alg-base;
    if-feature "ssh-dh-group-exchange";
    description
        "Diffie-Hellman Group and Key Exchange with SHA-1 as HASH.";
    reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
        Secure Shell (SSH) Transport Layer Protocol";
}

identity diffie-hellman-group-exchange-sha256 {
    base key-exchange-alg-base;
    if-feature "ssh-dh-group-exchange and ssh-sha2";
    description
        "Diffie-Hellman Group and Key Exchange with SHA-256 as HASH.";
    reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
        Secure Shell (SSH) Transport Layer Protocol";
}

identity ecdh-sha2-nistp256 {
    base key-exchange-alg-base;
    if-feature "ssh-ecc and ssh-sha2";
    description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
        nistp256 curve and the SHA2 family of hashing algorithms.";
```



```
    reference
      "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdh-sha2-nistp384 {
    base key-exchange-alg-base;
    if-feature "ssh-ecc and ssh-sha2";
    description
      "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
        nistp384 curve and the SHA2 family of hashing algorithms.";
    reference
      "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity ecdh-sha2-nistp521 {
    base key-exchange-alg-base;
    if-feature "ssh-ecc and ssh-sha2";
    description
      "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
        nistp521 curve and the SHA2 family of hashing algorithms.";
    reference
      "RFC 5656: Elliptic Curve Algorithm Integration in the
        Secure Shell Transport Layer";
  }

  identity encryption-alg-base {
    description
      "Base identity used to identify encryption algorithms.";
  }

  identity triple-des-cbc {
    base encryption-alg-base;
    description
      "Three-key 3DES in CBC mode.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }

  identity aes128-cbc {
    base encryption-alg-base;
    description
      "AES in CBC mode, with a 128-bit key.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  }
```

```
identity aes192-cbc {
  base encryption-alg-base;
  description
    "AES in CBC mode, with a 192-bit key.";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

identity aes256-cbc {
  base encryption-alg-base;
  description
    "AES in CBC mode, with a 256-bit key.";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

identity aes128-ctr {
  base encryption-alg-base;
  if-feature "ssh-ctr";
  description
    "AES in SDCTR mode, with 128-bit key.";
  reference
    "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
      Modes";
}

identity aes192-ctr {
  base encryption-alg-base;
  if-feature "ssh-ctr";
  description
    "AES in SDCTR mode, with 192-bit key.";
  reference
    "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
      Modes";
}

identity aes256-ctr {
  base encryption-alg-base;
  if-feature "ssh-ctr";
  description
    "AES in SDCTR mode, with 256-bit key.";
  reference
    "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
      Modes";
}

identity mac-alg-base {
  description
```

```
    "Base identity used to identify message authentication
      code (MAC) algorithms.";
}

identity hmac-sha1 {
  base mac-alg-base;
  description
    "HMAC-SHA1";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
}

identity hmac-sha2-256 {
  base mac-alg-base;
  if-feature "ssh-sha2";
  description
    "HMAC-SHA2-256";
  reference
    "RFC 6668: SHA-2 Data Integrity Verification for the
      Secure Shell (SSH) Transport Layer Protocol";
}

identity hmac-sha2-512 {
  base mac-alg-base;
  if-feature "ssh-sha2";
  description
    "HMAC-SHA2-512";
  reference
    "RFC 6668: SHA-2 Data Integrity Verification for the
      Secure Shell (SSH) Transport Layer Protocol";
}

// Groupings

grouping transport-params-grouping {
  description
    "A reusable grouping for SSH transport parameters.";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  container host-key {
    description
      "Parameters regarding host key.";
    leaf-list host-key-alg {
      type identityref {
        base public-key-alg-base;
      }
      ordered-by user;
      description

```

```
    "Acceptable host key algorithms in order of descending
      preference. The configured host key algorithms should
      be compatible with the algorithm used by the configured
      private key. Please see Section 5 of RFC XXXX for
      valid combinations.

      If this leaf-list is not configured (has zero elements)
      the acceptable host key algorithms are implementation-
      defined.";
  reference
    "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
}
}
container key-exchange {
  description
    "Parameters regarding key exchange.";
  leaf-list key-exchange-alg {
    type identityref {
      base key-exchange-alg-base;
    }
    ordered-by user;
    description
      "Acceptable key exchange algorithms in order of descending
      preference.

      If this leaf-list is not configured (has zero elements)
      the acceptable key exchange algorithms are implementation
      defined.";
  }
}
container encryption {
  description
    "Parameters regarding encryption.";
  leaf-list encryption-alg {
    type identityref {
      base encryption-alg-base;
    }
    ordered-by user;
    description
      "Acceptable encryption algorithms in order of descending
      preference.

      If this leaf-list is not configured (has zero elements)
      the acceptable encryption algorithms are implementation
      defined.";
  }
}
container mac {
```

```
description
  "Parameters regarding message authentication code (MAC).";
leaf-list mac-alg {
  type identityref {
    base mac-alg-base;
  }
  ordered-by user;
  description
    "Acceptable MAC algorithms in order of descending
     preference.

     If this leaf-list is not configured (has zero elements)
     the acceptable MAC algorithms are implementation-
     defined.";
}
}
}
}
<CODE ENDS>
```

## 6. Security Considerations

The YANG modules defined in this document are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the modules defined in this document define only groupings, these considerations are primarily for the designers of other modules that use these groupings.

There are a number of data nodes defined in the YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /: The entire data tree defined by all the modules defined in this draft are sensitive to write operations. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., can dramatically alter the implemented

security policy. However, no NACM annotations are applied as the data SHOULD be editable by users other than a designated 'recovery session'.

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

/client-auth/password: This node in the 'ietf-ssh-client' module is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. The only time this node should be returned is to support backup/restore type workflows. However, no NACM annotations are applied as the data SHOULD be writable by users other than a designated 'recovery session'.

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

NONE

## 7. IANA Considerations

### 7.1. The IETF XML Registry

This document registers three 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-ssh-client  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-server  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ssh-common  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

## 7.2. The YANG Module Names Registry

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

name:	ietf-ssh-client
namespace:	urn:ietf:params:xml:ns:yang:ietf-ssh-client
prefix:	sshc
reference:	RFC XXXX
name:	ietf-ssh-server
namespace:	urn:ietf:params:xml:ns:yang:ietf-ssh-server
prefix:	sshs
reference:	RFC XXXX
name:	ietf-ssh-common
namespace:	urn:ietf:params:xml:ns:yang:ietf-ssh-common
prefix:	sshcmn
reference:	RFC XXXX

## 8. References

### 8.1. Normative References

- [I-D.ietf-netconf-crypto-types]  
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-02 (work in progress), October 2018.
- [I-D.ietf-netconf-keystore]  
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-08 (work in progress), March 2019.
- [I-D.ietf-netconf-trust-anchors]  
Watsen, K., "YANG Data Model for Global Trust Anchors", draft-ietf-netconf-trust-anchors-03 (work in progress), March 2019.
- [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>>.

- [RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <<https://www.rfc-editor.org/info/rfc4344>>.
- [RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <<https://www.rfc-editor.org/info/rfc4419>>.
- [RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <<https://www.rfc-editor.org/info/rfc5656>>.
- [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>>.
- [RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <<https://www.rfc-editor.org/info/rfc6187>>.
- [RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <<https://www.rfc-editor.org/info/rfc6668>>.
- [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>>.
- [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>>.
- [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>>.

## 8.2. Informative References

- [OPENSSH] Project, T. O., "OpenSSH", 2016, <<http://www.openssh.com>>.



- [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>>.
- [RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <<https://www.rfc-editor.org/info/rfc4253>>.
- [RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, DOI 10.17487/RFC4254, January 2006, <<https://www.rfc-editor.org/info/rfc4254>>.
- [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>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <<https://www.rfc-editor.org/info/rfc7317>>.
- [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>>.
- [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>>.

## Appendix A. Change Log

## A.1. 00 to 01

- o Noted that '0.0.0.0' and ':::' might have special meanings.
- o Renamed "keychain" to "keystore".

## A.2. 01 to 02

- o Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.
- o Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.
- o Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

## A.3. 02 to 03

- o Removed 'RESTRICTED' enum from 'password' leaf type.
- o Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.
- o Fixed description statement for leaf 'trusted-ca-certs'.

## A.4. 03 to 04

- o Change title to "YANG Groupings for SSH Clients and SSH Servers"
- o Added reference to RFC 6668
- o Added RFC 8174 to Requirements Language Section.
- o Enhanced description statement for ietf-ssh-server's "trusted-ca-certs" leaf.
- o Added mandatory true to ietf-ssh-client's "client-auth" 'choice' statement.
- o Changed the YANG prefix for module ietf-ssh-common from 'sshcom' to 'sshcmn'.
- o Removed the compression algorithms as they are not commonly configurable in vendors' implementations.

- o Updating descriptions in transport-params-grouping and the servers's usage of it.
- o Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- o Updated YANG to use typedefs around leafrefs to common keystore paths
- o Now inlines key and certificates (no longer a leafref to keystore)

A.5. 04 to 05

- o Merged changes from co-author.

A.6. 05 to 06

- o Updated to use trust anchors from trust-anchors draft (was keystore draft)
- o Now uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

A.7. 06 to 07

- o factored the ssh-[client|server]-groupings into more reusable groupings.
- o added if-feature statements for the new "ssh-host-keys" and "x509-certificates" features defined in draft-ietf-netconf-trust-anchors.

A.8. 07 to 08

- o Added a number of compatibility matrices to Section 5 (thanks Frank!)
- o Clarified that any configured "host-key-alg" values need to be compatible with the configured private key.

A.9. 08 to 09

- o Updated examples to reflect update to groupings defined in the keystore -09 draft.
- o Add SSH keepalives features and groupings.
- o Prefixed top-level SSH grouping nodes with 'ssh-' and support mashups.

- o Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10

- o Reformatted the YANG modules.

A.11. 10 to 11

- o Reformatted lines causing folding to occur.

#### Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Michal Vasko, and Bert Wijnen.

#### Authors' Addresses

Kent Watsen  
Watsen Networks

EMail: kent+ietf@watsen.net

Gary Wu  
Cisco Systems

EMail: garywu@cisco.com

Liang Xia  
Huawei

EMail: frank.xialiang@huawei.com

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
G. Wu  
Cisco Systems  
L. Xia  
Huawei  
March 9, 2019

YANG Groupings for TLS Clients and TLS Servers  
draft-ietf-netconf-tls-client-server-10

Abstract

This document defines three YANG modules: the first defines groupings for a generic TLS client, the second defines groupings for a generic TLS server, and the third defines common identities and groupings used by both the client and the server. It is intended that these groupings will be used by applications using the TLS protocol.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o I-D.ietf-netconf-trust-anchors
- o I-D.ietf-netconf-keystore

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

- o "XXXX" --> the assigned RFC value for this draft
- o "YYYY" --> the assigned RFC value for I-D.ietf-netconf-trust-anchors
- o "ZZZZ" --> the assigned RFC value for I-D.ietf-netconf-keystore

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

#### Status of This Memo

This Internet-Draft 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 September 10, 2019.

#### Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. The TLS Client Model . . . . .	4
3.1. Tree Diagram . . . . .	4
3.2. Example Usage . . . . .	5
3.3. YANG Module . . . . .	6
4. The TLS Server Model . . . . .	10
4.1. Tree Diagram . . . . .	10

4.2. Example Usage . . . . .	11
4.3. YANG Module . . . . .	13
5. The TLS Common Model . . . . .	17
5.1. Tree Diagram . . . . .	25
5.2. Example Usage . . . . .	25
5.3. YANG Module . . . . .	25
6. Security Considerations . . . . .	34
7. IANA Considerations . . . . .	35
7.1. The IETF XML Registry . . . . .	35
7.2. The YANG Module Names Registry . . . . .	35
8. References . . . . .	36
8.1. Normative References . . . . .	36
8.2. Informative References . . . . .	37
Appendix A. Change Log . . . . .	39
A.1. 00 to 01 . . . . .	39
A.2. 01 to 02 . . . . .	39
A.3. 02 to 03 . . . . .	39
A.4. 03 to 04 . . . . .	39
A.5. 04 to 05 . . . . .	40
A.6. 05 to 06 . . . . .	40
A.7. 06 to 07 . . . . .	40
A.8. 07 to 08 . . . . .	40
A.9. 08 to 09 . . . . .	40
A.10. 09 to 10 . . . . .	40
Acknowledgements . . . . .	41
Authors' Addresses . . . . .	41

## 1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines a grouping for a generic TLS client, the second defines a grouping for a generic TLS server, and the third defines identities and groupings common to both the client and the server (TLS is defined in [RFC5246]). It is intended that these groupings will be used by applications using the TLS protocol. For instance, these groupings could be used to help define the data model for an HTTPS [RFC2818] server or a NETCONF over TLS [RFC7589] based server.

The client and server YANG modules in this document each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "ssh-server-grouping" grouping for the TLS parts it provides, while adding data nodes for the TCP-level call-home configuration.

The modules defined in this document use groupings defined in [I-D.ietf-netconf-keystore] enabling keys to be either locally defined or a reference to globally configured values.

## 2. Terminology

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

## 3. The TLS Client Model

### 3.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-tls-client" module that does not have groupings expanded.

module: ietf-tls-client

```
grouping tls-client-grouping
  +---u client-identity-grouping
  +---u server-auth-grouping
  +---u hello-params-grouping
  +---u keepalives-grouping
grouping client-identity-grouping
  +-- tls-client-identity
    +-- (auth-type)?
      +--:(certificate)
        +-- certificate
          +---u client-identity-grouping
grouping server-auth-grouping
  +-- tls-server-auth
    +-- pinned-ca-certs?      ta:pinned-certificates-ref
    |   {ta:x509-certificates}?
    +-- pinned-server-certs?  ta:pinned-certificates-ref
    |   {ta:x509-certificates}?
grouping hello-params-grouping
  +-- tls-hello-params {tls-client-hello-params-config}?
    +---u hello-params-grouping
grouping keepalives-grouping
  +-- tls-keepalives {tls-client-keepalives}?
    +-- max-wait?          uint16
    +-- max-attempts?      uint8
```



### 3.2. Example Usage

This section presents two examples showing the `tls-client-grouping` populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 3 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following example configures the client identity using a local key:

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

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

  <!-- how this client will authenticate itself to the server -->
  <tls-client-identity>
    <certificate>
      <local-definition>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto\
\types">ct:rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </certificate>
  </tls-client-identity>

  <!-- which certificates will this client trust -->
  <tls-server-auth>
    <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinned-ca-c\
\erts>
    <pinned-server-certs>explicitly-trusted-server-certs</pinned-ser\
\ver-certs>
  </tls-server-auth>

  <tls-keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </tls-keepalives>

</tls-client>
```

The following example configures the client identity using a key from the keystore:

===== NOTE: '\!' line wrapping per BCP XX (RFC XXXX) =====

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

  <!-- how this client will authenticate itself to the server -->
  <tls-client-identity>
    <certificate>
      <keystore-reference>ex-rsa-cert</keystore-reference>
    </certificate>
  </tls-client-identity>

  <!-- which certificates will this client trust -->
  <tls-server-auth>
    <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinned-ca-c\
\erts>
    <pinned-server-certs>explicitly-trusted-server-certs</pinned-ser\
\ver-certs>
  </tls-server-auth>

  <tls-keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </tls-keepalives>

</tls-client>
```

### 3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-tls-client@2019-03-09.yang"
module ietf-tls-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix tlsc;

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-trust-anchors {
    prefix ta;
    reference
      "RFC YYYY: YANG Data Model for Global Trust Anchors";
  }
}
```

```
}

import ietf-keystore {
  prefix ks;
  reference
    "RFC ZZZZ: YANG Data Model for a 'Keystore' Mechanism";
}

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

contact
  "WG Web:    <http://datatracker.ietf.org/wg/netconf/>
  WG List:    <mailto:netconf@ietf.org>
  Author:     Kent Watsen <mailto:kent+ietf@watsen.net>
  Author:     Gary Wu <mailto:garywu@cisco.com>";

description
  "This module defines reusable groupings for TLS clients that
  can be used as a basis for specific TLSclient instances.

  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.

  Copyright (c) 2019 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
}
```

```
// Features

feature tls-client-hello-params-config {
  description
    "TLS hello message parameters are configurable on a TLS
    client.";
}

feature tls-client-keepalives {
  description
    "Per socket TLS keepalive parameters are configurable for
    TLS clients on the server implementing this feature.";
}

// Groupings

grouping tls-client-grouping {
  description
    "A reusable grouping for configuring a TLS client without
    any consideration for how an underlying TCP session is
    established.";
  uses client-identity-grouping;
  uses server-auth-grouping;
  uses hello-params-grouping;
  uses keepalives-grouping;
}

grouping client-identity-grouping {
  description
    "A reusable grouping for configuring a TLS client identity.";
  container tls-client-identity {
    description
      "The credentials used by the client to authenticate to
      the TLS server.";
    choice auth-type {
      description
        "The authentication type.";
      container certificate {
        uses
          ks:local-or-keystore-end-entity-cert-with-key-grouping;
        description
          "A locally-defined or referenced certificate
          to be used for client authentication.";
        reference
          "RFC ZZZZ: YANG Data Model for a 'Keystore' Mechanism";
      }
    }
  }
}
```

```
}

grouping server-auth-grouping {
  description
    "A reusable grouping for configuring TLS server
    authentication.";
  container tls-server-auth {
    must 'pinned-ca-certs or pinned-server-certs';
    description
      "Trusted server identities.";
    leaf pinned-ca-certs {
      if-feature "ta:x509-certificates";
      type ta:pinned-certificates-ref;
      description
        "A reference to a list of certificate authority (CA)
        certificates used by the TLS client to authenticate
        TLS server certificates. A server certificate is
        authenticated if it has a valid chain of trust to
        a configured pinned CA certificate.";
    }
    leaf pinned-server-certs {
      if-feature "ta:x509-certificates";
      type ta:pinned-certificates-ref;
      description
        "A reference to a list of server certificates used by
        the TLS client to authenticate TLS server certificates.
        A server certificate is authenticated if it is an
        exact match to a configured pinned server certificate.";
    }
  }
}

grouping hello-params-grouping {
  description
    "A reusable grouping for configuring a TLS transport
    parameters.";
  container tls-hello-params {
    if-feature "tls-client-hello-params-config";
    uses tlscmn:hello-params-grouping;
    description
      "Configurable parameters for the TLS hello message.";
  }
}

grouping keepalives-grouping {
  description
    "A reusable grouping for configuring TLS client keepalive
    parameters.";
```

```
container tls-keepalives {
  if-feature "tls-client-keepalives";
  description
    "Configures the keep-alive policy, to proactively test
     the aliveness of the TLS server.  An unresponsive
     TLS server is dropped after approximately max-wait
     * max-attempts seconds.";
  leaf max-wait {
    type uint16 {
      range "1..max";
    }
    units "seconds";
    default "30";
    description
      "Sets the amount of time in seconds after which if no data
       has been received from the TLS server, a TLS-level message
       will be sent to test the aliveness of the TLS server.";
  }
  leaf max-attempts {
    type uint8;
    default "3";
    description
      "Sets the maximum number of sequential keep-alive messages
       that can fail to obtain a response from the TLS server
       before assuming the TLS server is no longer alive.";
  }
}
}
}
}
<CODE ENDS>
```

## 4. The TLS Server Model

### 4.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-tls-server" module that does not have groupings expanded.

```
module: ietf-tls-server

grouping tls-server-grouping
  +---u server-identity-grouping
  +---u client-auth-grouping
  +---u hello-params-grouping
  +---u keepalives-grouping
grouping server-identity-grouping
  +-- tls-server-identity
  +---u server-identity-grouping
grouping client-auth-grouping
  +-- tls-client-auth
  +-- pinned-ca-certs?      ta:pinned-certificates-ref
  |   {ta:x509-certificates}?
  +-- pinned-client-certs?  ta:pinned-certificates-ref
  |   {ta:x509-certificates}?
grouping hello-params-grouping
  +-- tls-hello-params {tls-server-hello-params-config}?
  +---u hello-params-grouping
grouping keepalives-grouping
  +-- tls-keepalives {tls-server-keepalives}?
  +-- max-wait?      uint16
  +-- max-attempts?  uint8
```

#### 4.2. Example Usage

This section presents two examples showing the `tls-server-grouping` populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 3 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

The following example configures the server identity using a local key:

===== NOTE: '\\\\' line wrapping per BCP XX (RFC XXXX) =====

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

  <!-- how this server will authenticate itself to the client -->
  <tls-server-identity>
    <local-definition>
      <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-t\
ypes">ct:rsa2048</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
      <cert>base64encodedvalue==</cert>
    </local-definition>
  </tls-server-identity>

  <!-- which certificates will this server trust -->
  <tls-client-auth>
    <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
erts>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
ent-certs>
  </tls-client-auth>

</tls-server>
```

The following example configures the server identity using a key from the keystore:

===== NOTE: '\\\\' line wrapping per BCP XX (RFC XXXX) =====

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

  <!-- how this server will authenticate itself to the client -->
  <tls-server-identity>
    <keystore-reference>ex-rsa-cert</keystore-reference>
  </tls-server-identity>

  <!-- which certificates will this server trust -->
  <tls-client-auth>
    <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
erts>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
ent-certs>
  </tls-client-auth>

</tls-server>
```



#### 4.3. YANG Module

This YANG module has a normative references to [RFC5246], [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-tls-server@2019-03-09.yang"
module ietf-tls-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix tlss;

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2019-03-09; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-trust-anchors {
    prefix ta;
    reference
      "RFC YYYY: YANG Data Model for Global Trust Anchors";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC ZZZZ: YANG Data Model for a 'Keystore' Mechanism";
  }

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

  contact
    "WG Web:    <http://datatracker.ietf.org/wg/netconf/>
    WG List:    <mailto:netconf@ietf.org>
    Author:     Kent Watsen <mailto:kent+ietf@watsen.net>
    Author:     Gary Wu <mailto:garywu@cisco.com>";

  description
    "This module defines reusable groupings for TLS servers that
    can be used as a basis for specific TLS server instances.

    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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
}

// Features

feature tls-server-hello-params-config {
  description
    "TLS hello message parameters are configurable on a TLS
    server.";
}

feature tls-server-keepalives {
  description
    "Per socket TLS keepalive parameters are configurable for
    TLS servers on the server implementing this feature.";
}

// Groupings

grouping tls-server-grouping {
  description
    "A reusable grouping for configuring a TLS server without
    any consideration for how underlying TCP sessions are
    established.";
  uses server-identity-grouping;
  uses client-auth-grouping;
  uses hello-params-grouping;
  uses keepalives-grouping;
}
```

```
grouping server-identity-grouping {
  description
    "A reusable grouping for configuring a TLS server identity.";
  container tls-server-identity {
    description
      "A locally-defined or referenced end-entity certificate,
       including any configured intermediate certificates, the
       TLS server will present when establishing a TLS connection
       in its Certificate message, as defined in Section 7.4.2
       in RFC 5246.";
    reference
      "RFC 5246:
       The Transport Layer Security (TLS) Protocol Version 1.2
       RFC ZZZZ:
       YANG Data Model for a 'Keystore' Mechanism";
    uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
  }
}

grouping client-auth-grouping {
  description
    "A reusable grouping for configuring a TLS client
    authentication.";
  container tls-client-auth {
    description
      "A reference to a list of pinned certificate authority (CA)
       certificates and a reference to a list of pinned client
       certificates.";
    leaf pinned-ca-certs {
      if-feature "ta:x509-certificates";
      type ta:pinned-certificates-ref;
      description
        "A reference to a list of certificate authority (CA)
         certificates used by the TLS server to authenticate
         TLS client certificates. A client certificate is
         authenticated if it has a valid chain of trust to
         a configured pinned CA certificate.";
      reference
        "RFC YYYY: YANG Data Model for Global Trust Anchors";
    }
    leaf pinned-client-certs {
      if-feature "ta:x509-certificates";
      type ta:pinned-certificates-ref;
      description
        "A reference to a list of client certificates used by
         the TLS server to authenticate TLS client certificates.
         A clients certificate is authenticated if it is an
         exact match to a configured pinned client certificate.";
```

```
        reference
          "RFC YYYY: YANG Data Model for Global Trust Anchors";
      }
    }
  }

  grouping hello-params-grouping {
    description
      "A reusable grouping for configuring a TLS transport
      parameters.";
    container tls-hello-params {
      if-feature "tls-server-hello-params-config";
      uses tlscmn:hello-params-grouping;
      description
        "Configurable parameters for the TLS hello message.";
    }
  }

  grouping keepalives-grouping {
    description
      "A reusable grouping for configuring TLS server keepalive
      parameters.";
    container tls-keepalives {
      if-feature "tls-server-keepalives";
      description
        "Configures the keep-alive policy, to proactively test
        the aliveness of the TLS client.  An unresponsive
        TLS client is dropped after approximately max-wait
        * max-attempts seconds.";
      leaf max-wait {
        type uint16 {
          range "1..max";
        }
        units "seconds";
        default "30";
        description
          "Sets the amount of time in seconds after which if no data
          has been received from the TLS client, a TLS-level message
          will be sent to test the aliveness of the TLS client.";
      }
      leaf max-attempts {
        type uint8;
        default "3";
        description
          "Sets the maximum number of sequential keep-alive messages
          that can fail to obtain a response from the TLS client
          before assuming the TLS client is no longer alive.";
      }
    }
  }
```

```
    }  
  }  
}  
<CODE ENDS>
```

## 5. The TLS Common Model

The TLS common model presented in this section contains identities and groupings common to both TLS clients and TLS servers. The hello-params-grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for TLS clients and TLS servers that are capable of doing so and may serve to make TLS clients and TLS servers compliant with local security policies. This model supports both TLS1.2 [RFC5246] and TLS 1.3 [RFC8446].

TLS 1.2 and TLS 1.3 have different ways defining their own supported cryptographic algorithms, see TLS and DTLS IANA registries page (<https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml>):

- o TLS 1.2 defines four categories of registries for cryptographic algorithms: TLS Cipher Suites, TLS SignatureAlgorithm, TLS HashAlgorithm, TLS Supported Groups. TLS Cipher Suites plays the role of combining all of them into one set, as each value of the set represents a unique and feasible combination of all the cryptographic algorithms, and thus the other three registry categories do not need to be considered here. In this document, the TLS common model only chooses those TLS1.2 algorithms in TLS Cipher Suites which are marked as recommended:  
TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256,  
TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384,  
TLS\_DHE\_PSK\_WITH\_AES\_128\_GCM\_SHA256,  
TLS\_DHE\_PSK\_WITH\_AES\_256\_GCM\_SHA384, and so on. All chosen algorithms are enumerated in Table 1-1 below;
- o TLS 1.3 defines its supported algorithms differently. Firstly, it defines three categories of registries for cryptographic algorithms: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Secondly, all three of these categories are useful, since they represent different parts of all the supported algorithms respectively. Thus, all of these registries categories are considered here. In this draft, the TLS common model chooses only

those TLS1.3 algorithms specified in B.4, 4.2.3, 4.2.7 of [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the hello-params-grouping should include three parameters for configuring its permitted TLS algorithms, which are: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Note that TLS1.2 only uses TLS Cipher Suites.

[I-D.ietf-netconf-crypto-types] defines six categories of cryptographic algorithms (hash-algorithm, symmetric-key-encryption-algorithm, mac-algorithm, asymmetric-key-encryption-algorithm, signature-algorithm, key-negotiation-algorithm) and lists several widely accepted algorithms for each of them. The TLS client and server models use one or more of these algorithms. The following tables are provided, in part to define the subset of algorithms defined in the crypto-types model used by TLS, and in part to ensure compatibility of configured TLS cryptographic parameters for configuring its permitted TLS algorithms:

ciper-suites in hello-params-grouping	HASH
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	sha-256
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_DHE_RSA_WITH_AES_128_CCM	sha-256
TLS_DHE_RSA_WITH_AES_256_CCM	sha-256
TLS_DHE_PSK_WITH_AES_128_CCM	sha-256
TLS_DHE_PSK_WITH_AES_256_CCM	sha-256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	sha-256

Table 1-1 TLS 1.2 Compatibility Matrix Part 1: ciper-suites mapping to hash-algorithm

ciper-suites in hello-params-grouping	symmetric
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_DHE_RSA_WITH_AES_128_CCM	enc-aes-128-ccm
TLS_DHE_RSA_WITH_AES_256_CCM	enc-aes-256-ccm
TLS_DHE_PSK_WITH_AES_128_CCM	enc-aes-128-ccm
TLS_DHE_PSK_WITH_AES_256_CCM	enc-aes-256-ccm
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	enc-aes-128-ccm

Table 1-2 TLS 1.2 Compatibility Matrix Part 2: ciper-suites mapping to symmetric-key-encryption-algorithm

ciper-suites in hello-params-grouping	MAC
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_DHE_RSA_WITH_AES_128_CCM	mac-aes-128-ccm
TLS_DHE_RSA_WITH_AES_256_CCM	mac-aes-256-ccm
TLS_DHE_PSK_WITH_AES_128_CCM	mac-aes-128-ccm
TLS_DHE_PSK_WITH_AES_256_CCM	mac-aes-256-ccm
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	mac-aes-128-ccm

Table 1-3 TLS 1.2 Compatibility Matrix Part 3: ciper-suites mapping to MAC-algorithm



ciper-suites in hello-params-grouping	signature
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	rsa-pkcs1-sha256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	rsa-pkcs1-sha384
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	N/A
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	N/A
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ecdsa-secp256r1-sha256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ecdsa-secp384r1-sha384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	rsa-pkcs1-sha384
TLS_DHE_RSA_WITH_AES_128_CCM	rsa-pkcs1-sha256
TLS_DHE_RSA_WITH_AES_256_CCM	rsa-pkcs1-sha256
TLS_DHE_PSK_WITH_AES_128_CCM	N/A
TLS_DHE_PSK_WITH_AES_256_CCM	N/A
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	ecdsa-secp256r1-sha256
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	N/A
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	N/A
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	N/A
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	N/A
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	N/A

Table 1-4 TLS 1.2 Compatibility Matrix Part 4: ciper-suites mapping to signature-algorithm

ciper-suites in hello-params-grouping	key-negotiation
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	dhe-ffdhe2048, ...
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	psk-dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	psk-dhe-ffdhe2048, ...
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ecdhe-secp256r1, ...
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ecdhe-secp256r1, ...
TLS_DHE_RSA_WITH_AES_128_CCM	dhe-ffdhe2048, ...
TLS_DHE_RSA_WITH_AES_256_CCM	dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_128_CCM	psk-dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_256_CCM	psk-dhe-ffdhe2048, ...
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	ecdhe-secp256r1, ...
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	dhe-ffdhe2048, ...
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	psk-ecdhe-secp256r1,...
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	psk-dhe-ffdhe2048, ...
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	psk-ecdhe-secp256r1,...
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	psk-ecdhe-secp256r1,...
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	psk-ecdhe-secp256r1,...

Table 1-5 TLS 1.2 Compatibility Matrix Part 5: ciper-suites mapping to key-negotiation-algorithm

ciper-suites in hello-params-grouping	HASH
TLS_AES_128_GCM_SHA256	sha-256
TLS_AES_256_GCM_SHA384	sha-384
TLS_CHACHA20_POLY1305_SHA256	sha-256
TLS_AES_128_CCM_SHA256	sha-256

Table 2-1 TLS 1.3 Compatibility Matrix Part 1: ciper-suites mapping to hash-algorithm

ciper-suites in hello -params-grouping	symmetric
TLS_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_AES_256_GCM_SHA384	enc-aes-128-gcm
TLS_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_AES_128_CCM_SHA256	enc-aes-128-ccm

Table 2-2 TLS 1.3 Compatibility Matrix Part 2: ciper-suites mapping to symmetric-key--encryption-algorithm

ciper-suites in hello -params-grouping	symmetric
TLS_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_AES_256_GCM_SHA384	mac-aes-128-gcm
TLS_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_AES_128_CCM_SHA256	mac-aes-128-ccm

Table 2-3 TLS 1.3 Compatibility Matrix Part 3: ciper-suites mapping to MAC-algorithm

signatureScheme in hello -params-grouping	signature
rsa-pkcs1-sha256	rsa-pkcs1-sha256
rsa-pkcs1-sha384	rsa-pkcs1-sha384
rsa-pkcs1-sha512	rsa-pkcs1-sha512
rsa-pss-rsae-sha256	rsa-pss-rsae-sha256
rsa-pss-rsae-sha384	rsa-pss-rsae-sha384
rsa-pss-rsae-sha512	rsa-pss-rsae-sha512
rsa-pss-pss-sha256	rsa-pss-pss-sha256
rsa-pss-pss-sha384	rsa-pss-pss-sha384
rsa-pss-pss-sha512	rsa-pss-pss-sha512
ecdsa-secp256r1-sha256	ecdsa-secp256r1-sha256
ecdsa-secp384r1-sha384	ecdsa-secp384r1-sha384
ecdsa-secp521r1-sha512	ecdsa-secp521r1-sha512
ed25519	ed25519
ed448	ed448

Table 2-4 TLS 1.3 Compatibility Matrix Part 4: SignatureScheme mapping to signature-algorithm

supported Groups in hello -params-grouping	key-negotiation
dhe-ffdhe2048	dhe-ffdhe2048
dhe-ffdhe3072	dhe-ffdhe3072
dhe-ffdhe4096	dhe-ffdhe4096
dhe-ffdhe6144	dhe-ffdhe6144
dhe-ffdhe8192	dhe-ffdhe8192
psk-dhe-ffdhe2048	psk-dhe-ffdhe2048
psk-dhe-ffdhe3072	psk-dhe-ffdhe3072
psk-dhe-ffdhe4096	psk-dhe-ffdhe4096
psk-dhe-ffdhe6144	psk-dhe-ffdhe6144
psk-dhe-ffdhe8192	psk-dhe-ffdhe8192
ecdhe-secp256r1	ecdhe-secp256r1
ecdhe-secp384r1	ecdhe-secp384r1
ecdhe-secp521r1	ecdhe-secp521r1
ecdhe-x25519	ecdhe-x25519
ecdhe-x448	ecdhe-x448
psk-ecdhe-secp256r1	psk-ecdhe-secp256r1
psk-ecdhe-secp384r1	psk-ecdhe-secp384r1
psk-ecdhe-secp521r1	psk-ecdhe-secp521r1
psk-ecdhe-x25519	psk-ecdhe-x25519
psk-ecdhe-x448	psk-ecdhe-x448

Table 2-5 TLS 1.3 Compatibility Matrix Part 5: Supported Groups mapping to key-negotiation-algorithm

Note that in Table 1-5:

- o dhe-ffdhe2048, ... is the abbreviation of dhe-ffdhe2048, dhe-ffdhe3072, dhe-ffdhe4096, dhe-ffdhe6144, dhe-ffdhe8192;
- o psk-dhe-ffdhe2048, ... is the abbreviation of psk-dhe-ffdhe2048, psk-dhe-ffdhe3072, psk-dhe-ffdhe4096, psk-dhe-ffdhe6144, psk-dhe-ffdhe8192;
- o ecdhe-secp256r1, ... is the abbreviation of ecdhe-secp256r1, ecdhe-secp384r1, ecdhe-secp521r1, ecdhe-x25519, ecdhe-x448;
- o psk-ecdhe-secp256r1, ... is the abbreviation of psk-ecdhe-secp256r1, psk-ecdhe-secp384r1, psk-ecdhe-secp521r1, psk-ecdhe-x25519, psk-ecdhe-x448.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive.

### 5.1. Tree Diagram

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-tls-common" module.

```
module: ietf-tls-common

  grouping hello-params-grouping
    +-- tls-versions
    |   +-- tls-version*   identityref
    +-- cipher-suites
    |   +-- cipher-suite*   identityref
```

### 5.2. Example Usage

This section shows how it would appear if the transport-params-grouping were populated with some data.

```
<hello-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common"
  xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
  <tls-versions>
    <tls-version>tlscmn:tls-1.1</tls-version>
    <tls-version>tlscmn:tls-1.2</tls-version>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>tlscmn:dhe-rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>tlscmn:rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>tlscmn:rsa-with-3des-edc-cbc-sha</cipher-suite>
  </cipher-suites>
</hello-params>
```

### 5.3. YANG Module

This YANG module has a normative references to [RFC4346], [RFC5246], [RFC5288], [RFC5289], and [RFC8422].

This YANG module has a informative references to [RFC2246], [RFC4346], [RFC5246], and [RFC8446].

```
<CODE BEGINS> file "ietf-tls-common@2019-03-09.yang"
module ietf-tls-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
  prefix tlscmn;

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

## contact

"WG Web: <<http://datatracker.ietf.org/wg/netconf/>>  
WG List: <<mailto:netconf@ietf.org>>  
Author: Kent Watsen <<mailto:kent+ietf@watsen.net>>  
Author: Gary Wu <<mailto:garywu@cisco.com>>"

## description

"This module defines a common features, identities, and groupings for Transport Layer Security (TLS).

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";  
}  
  
// Features  
  
feature tls-1_0 {  
  description  
    "TLS Protocol Version 1.0 is supported.";  
  reference  
    "RFC 2246: The TLS Protocol Version 1.0";  
}  
  
feature tls-1_1 {  
  description
```

```
        "TLS Protocol Version 1.1 is supported.";
    reference
        "RFC 4346: The Transport Layer Security (TLS) Protocol
          Version 1.1";
}

feature tls-1_2 {
    description
        "TLS Protocol Version 1.2 is supported.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
}

feature tls-1_3 {
    description
        "TLS Protocol Version 1.2 is supported.";
    reference
        "RFC 8446: The Transport Layer Security (TLS) Protocol
          Version 1.3";
}

feature tls-ecc {
    description
        "Elliptic Curve Cryptography (ECC) is supported for TLS.";
    reference
        "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
          for Transport Layer Security (TLS)";
}

feature tls-dhe {
    description
        "Ephemeral Diffie-Hellman key exchange is supported for TLS.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
}

feature tls-3des {
    description
        "The Triple-DES block cipher is supported for TLS.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
}

feature tls-gcm {
    description
```

```
    "The Galois/Counter Mode authenticated encryption mode is
      supported for TLS.";
  reference
    "RFC 5288: AES Galois Counter Mode (GCM) Cipher Suites for
      TLS";
}

feature tls-sha2 {
  description
    "The SHA2 family of cryptographic hash functions is supported
      for TLS.";
  reference
    "FIPS PUB 180-4: Secure Hash Standard (SHS)";
}

// Identities

identity tls-version-base {
  description
    "Base identity used to identify TLS protocol versions.";
}

identity tls-1.0 {
  base tls-version-base;
  if-feature "tls-1_0";
  description
    "TLS Protocol Version 1.0.";
  reference
    "RFC 2246: The TLS Protocol Version 1.0";
}

identity tls-1.1 {
  base tls-version-base;
  if-feature "tls-1_1";
  description
    "TLS Protocol Version 1.1.";
  reference
    "RFC 4346: The Transport Layer Security (TLS) Protocol
      Version 1.1";
}

identity tls-1.2 {
  base tls-version-base;
  if-feature "tls-1_2";
  description
    "TLS Protocol Version 1.2.";
  reference
    "RFC 5246: The Transport Layer Security (TLS) Protocol
```



```
        Version 1.2";
    }

    identity cipher-suite-base {
        description
            "Base identity used to identify TLS cipher suites.";
    }

    identity rsa-with-aes-128-cbc-sha {
        base cipher-suite-base;
        description
            "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA.";
        reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
             Version 1.2";
    }

    identity rsa-with-aes-256-cbc-sha {
        base cipher-suite-base;
        description
            "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA.";
        reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
             Version 1.2";
    }

    identity rsa-with-aes-128-cbc-sha256 {
        base cipher-suite-base;
        if-feature "tls-sha2";
        description
            "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA256.";
        reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
             Version 1.2";
    }

    identity rsa-with-aes-256-cbc-sha256 {
        base cipher-suite-base;
        if-feature "tls-sha2";
        description
            "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA256.";
        reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
             Version 1.2";
    }

    identity dhe-rsa-with-aes-128-cbc-sha {
        base cipher-suite-base;
```

```
    if-feature "tls-dhe";
    description
        "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
}

identity dhe-rsa-with-aes-256-cbc-sha {
    base cipher-suite-base;
    if-feature "tls-dhe";
    description
        "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
}

identity dhe-rsa-with-aes-128-cbc-sha256 {
    base cipher-suite-base;
    if-feature "tls-dhe and tls-sha2";
    description
        "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA256.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
}

identity dhe-rsa-with-aes-256-cbc-sha256 {
    base cipher-suite-base;
    if-feature "tls-dhe and tls-sha2";
    description
        "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA256.";
    reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
          Version 1.2";
}

identity ecdhe-ecdsa-with-aes-128-cbc-sha256 {
    base cipher-suite-base;
    if-feature "tls-ecc and tls-sha2";
    description
        "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256.";
    reference
        "RFC 5289: TLS Elliptic Curve Cipher Suites with
          SHA-256/384 and AES Galois Counter Mode (GCM)";
}
```

```
identity ecdhe-ecdsa-with-aes-256-cbc-sha384 {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-sha2";
  description
    "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-rsa-with-aes-128-cbc-sha256 {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-sha2";
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-rsa-with-aes-256-cbc-sha384 {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-sha2";
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-ecdsa-with-aes-128-gcm-sha256 {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-gcm and tls-sha2";
  description
    "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-ecdsa-with-aes-256-gcm-sha384 {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-gcm and tls-sha2";
  description
    "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}
```

```
}

identity ecdhe-rsa-with-aes-128-gcm-sha256 {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-gcm and tls-sha2";
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity ecdhe-rsa-with-aes-256-gcm-sha384 {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-gcm and tls-sha2";
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.";
  reference
    "RFC 5289: TLS Elliptic Curve Cipher Suites with
      SHA-256/384 and AES Galois Counter Mode (GCM)";
}

identity rsa-with-3des-ede-cbc-sha {
  base cipher-suite-base;
  if-feature "tls-3des";
  description
    "Cipher suite TLS_RSA_WITH_3DES_EDE_CBC_SHA.";
  reference
    "RFC 5246: The Transport Layer Security (TLS) Protocol
      Version 1.2";
}

identity ecdhe-rsa-with-3des-ede-cbc-sha {
  base cipher-suite-base;
  if-feature "tls-ecc and tls-3des";
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA.";
  reference
    "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
      for Transport Layer Security (TLS)";
}

identity ecdhe-rsa-with-aes-128-cbc-sha {
  base cipher-suite-base;
  if-feature "tls-ecc";
  description
    "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA.";
  reference
```

```
        "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
          for Transport Layer Security (TLS)";
    }

    identity ecdhe-rsa-with-aes-256-cbc-sha {
        base cipher-suite-base;
        if-feature "tls-ecc";
        description
            "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA.";
        reference
            "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
              for Transport Layer Security (TLS)";
    }

    // Groupings

    grouping hello-params-grouping {
        description
            "A reusable grouping for TLS hello message parameters.";
        reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
              Version 1.2";
        container tls-versions {
            description
                "Parameters regarding TLS versions.";
            leaf-list tls-version {
                type identityref {
                    base tls-version-base;
                }
            }
            description
                "Acceptable TLS protocol versions.

                If this leaf-list is not configured (has zero elements)
                the acceptable TLS protocol versions are implementation-
                defined.";
        }
    }

    container cipher-suites {
        description
            "Parameters regarding cipher suites.";
        leaf-list cipher-suite {
            type identityref {
                base cipher-suite-base;
            }
        }
        ordered-by user;
        description
            "Acceptable cipher suites in order of descending
              preference. The configured host key algorithms should
```

be compatible with the algorithm used by the configured private key. Please see Section 5 of RFC XXXX for valid combinations.

If this leaf-list is not configured (has zero elements) the acceptable cipher suites are implementation-defined.";

reference

"RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";

}

}

}

}

<CODE ENDS>

## 6. Security Considerations

The YANG modules defined in this document are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the modules defined in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

There are a number of data nodes defined in the YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

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

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

NONE

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

NONE

## 7. IANA Considerations

### 7.1. The IETF XML Registry

This document registers three 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-tls-client  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-server  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-common  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

### 7.2. The YANG Module Names Registry

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

name: ietf-tls-client  
namespace: urn:ietf:params:xml:ns:yang:ietf-tls-client  
prefix: tlsc  
reference: RFC XXXX

name: ietf-tls-server  
namespace: urn:ietf:params:xml:ns:yang:ietf-tls-server  
prefix: tlss  
reference: RFC XXXX

name: ietf-tls-common  
namespace: urn:ietf:params:xml:ns:yang:ietf-tls-common  
prefix: tlscmn  
reference: RFC XXXX

## 8. References

### 8.1. Normative References

- [I-D.ietf-netconf-crypto-types]  
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-02 (work in progress), October 2018.
- [I-D.ietf-netconf-keystore]  
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-08 (work in progress), March 2019.
- [I-D.ietf-netconf-trust-anchors]  
Watsen, K., "YANG Data Model for Global Trust Anchors", draft-ietf-netconf-trust-anchors-03 (work in progress), March 2019.
- [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>>.
- [RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <<https://www.rfc-editor.org/info/rfc5288>>.
- [RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <<https://www.rfc-editor.org/info/rfc5289>>.



- [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>>.
- [RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <<https://www.rfc-editor.org/info/rfc7589>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <<https://www.rfc-editor.org/info/rfc8422>>.
- [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>>.

## 8.2. Informative References

- [RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <<https://www.rfc-editor.org/info/rfc2246>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.

- [RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <<https://www.rfc-editor.org/info/rfc4346>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [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>>.
- [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>>.

## Appendix A. Change Log

## A.1. 00 to 01

- o Noted that '0.0.0.0' and ':::' might have special meanings.
- o Renamed "keychain" to "keystore".

## A.2. 01 to 02

- o Removed the groupings containing transport-level configuration. Now modules contain only the transport-independent groupings.
- o Filled in previously incomplete 'ietf-tls-client' module.
- o Added cipher suites for various algorithms into new 'ietf-tls-common' module.

## A.3. 02 to 03

- o Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.
- o Fixed description statement for leaf 'trusted-ca-certs'.

## A.4. 03 to 04

- o Updated title to "YANG Groupings for TLS Clients and TLS Servers"
- o Updated leafref paths to point to new keystore path
- o Changed the YANG prefix for ietf-tls-common from 'tlscom' to 'tlscmn'.
- o Added TLS protocol versions 1.0 and 1.1.
- o Made author lists consistent
- o Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- o Updated YANG to use typedefs around leafrefs to common keystore paths
- o Now inlines key and certificates (no longer a leafref to keystore)

## A.5. 04 to 05

- o Merged changes from co-author.

## A.6. 05 to 06

- o Updated to use trust anchors from trust-anchors draft (was keystore draft)
- o Now Uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

## A.7. 06 to 07

- o factored the tls-[client|server]-groupings into more reusable groupings.
- o added if-feature statements for the new "x509-certificates" feature defined in draft-ietf-netconf-trust-anchors.

## A.8. 07 to 08

- o Added a number of compatibility matrices to Section 5 (thanks Frank!)
- o Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

## A.9. 08 to 09

- o Updated examples to reflect update to groupings defined in the keystore draft.
- o Add TLS keepalives features and groupings.
- o Prefixed top-level TLS grouping nodes with 'tls-' and support mashups.
- o Updated copyright date, boilerplate template, affiliation, and folding algorithm.

## A.10. 09 to 10

- o Reformatted the YANG modules.

## Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, and Bert Wijnen.

## Authors' Addresses

Kent Watsen  
Watsen Networks

EMail: kent+ietf@watsen.net

Gary Wu  
Cisco Systems

EMail: garywu@cisco.com

Liang Xia  
Huawei

EMail: frank.xialiang@huawei.com

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
March 9, 2019

YANG Data Model for Global Trust Anchors  
draft-ietf-netconf-trust-anchors-03

Abstract

This document defines a YANG 1.1 data model for configuring global sets of X.509 certificates and SSH host-keys that can be referenced by other data models for trust. While the SSH host-keys are uniquely for the SSH protocol, the X.509 certificates may have multiple uses, including authenticating protocol peers and verifying signatures.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o "XXXX" --> the assigned RFC value for this draft
- o "YYYY" --> the assigned RFC value for draft-ietf-netconf-crypto-types

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

Status of This Memo

This Internet-Draft 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 September 10, 2019.

#### Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

#### Table of Contents

1. Introduction . . . . .	3
1.1. Requirements Language . . . . .	3
1.2. Tree Diagram Notation . . . . .	3
2. The Trust Anchors Model . . . . .	3
2.1. Tree Diagram . . . . .	3
2.2. Example Usage . . . . .	4
2.3. YANG Module . . . . .	7
3. Security Considerations . . . . .	11
4. IANA Considerations . . . . .	12
4.1. The IETF XML Registry . . . . .	12
4.2. The YANG Module Names Registry . . . . .	12
5. References . . . . .	13
5.1. Normative References . . . . .	13
5.2. Informative References . . . . .	13
Appendix A. Change Log . . . . .	15
A.1. 00 to 01 . . . . .	15
A.2. 01 to 02 . . . . .	15
A.3. 02 to 03 . . . . .	15
Acknowledgements . . . . .	15
Author's Address . . . . .	15

## 1. Introduction

This document defines a YANG 1.1 [RFC7950] data model for configuring global sets of X.509 certificates and SSH host-keys that can be referenced by other data models for trust. While the SSH host-keys are uniquely for the SSH protocol, the X.509 certificates may be used for multiple uses, including authenticating protocol peers and verifying signatures.

This document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, to support trust anchors installed during manufacturing, it is expected that such data may appear only in <operational>.

### 1.1. Requirements 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.

### 1.2. Tree Diagram Notation

Tree diagrams used in this document follow the notation defined in [RFC8340].

## 2. The Trust Anchors Model

### 2.1. Tree Diagram

The following tree diagram provides an overview of the "ietf-trust-anchors" module.



```

module: ietf-trust-anchors
  +--rw trust-anchors
    +--rw pinned-certificates* [name] {x509-certificates}?
      +--rw name string
      +--rw description? string
      +--rw pinned-certificate* [name]
        +--rw name string
        +--rw cert trust-anchor-cert-cms
        +---n certificate-expiration
          +-- expiration-date yang:date-and-time
    +--rw pinned-host-keys* [name] {ssh-host-keys}?
      +--rw name string
      +--rw description? string
      +--rw pinned-host-key* [name]
        +--rw name string
        +--rw host-key ct:ssh-host-key

```

## 2.2. Example Usage

The following example illustrates trust anchors in <operational> as described by Section 5.3 in [RFC8342]. This datastore view illustrates data set by the manufacturing process alongside conventional configuration. This trust anchors instance has six sets of pinned certificates and one set of pinned host keys.

```

<trust-anchors
  xmlns="urn:ietf:params:xml:ns:yang:ietf-trust-anchors"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">

  <!-- Manufacturer's trusted root CA certs -->
  <pinned-certificates or:origin="or:system">
    <name>manufacturers-root-ca-certs</name>
    <description>
      Certificates built into the device for authenticating
      manufacturer-signed objects, such as TLS server certificates,
      vouchers, etc. Note, though listed here, these are not
      configurable; any attempt to do so will be denied.
    </description>
    <pinned-certificate>
      <name>Manufacturer Root CA cert 1</name>
      <cert>base64encodedvalue==</cert>
    </pinned-certificate>
    <pinned-certificate>
      <name>Manufacturer Root CA cert 2</name>
      <cert>base64encodedvalue==</cert>
    </pinned-certificate>
  </pinned-certificates>

```

```
<!-- specific end-entity certs for authenticating servers -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-server-certs</name>
  <description>
    Specific server authentication certificates for explicitly
    trusted servers. These are needed for server certificates
    that are not signed by a pinned CA.
  </description>
  <pinned-certificate>
    <name>Fred Flintstone</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- trusted CA certs for authenticating servers -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-server-ca-certs</name>
  <description>
    Trust anchors (i.e. CA certs) that are used to authenticate
    server connections. Servers are authenticated if their
    certificate has a chain of trust to one of these CA
    certificates.
  </description>
  <pinned-certificate>
    <name>ca.example.com</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- specific end-entity certs for authenticating clients -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-client-certs</name>
  <description>
    Specific client authentication certificates for explicitly
    trusted clients. These are needed for client certificates
    that are not signed by a pinned CA.
  </description>
  <pinned-certificate>
    <name>George Jetson</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- trusted CA certs for authenticating clients -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-client-ca-certs</name>
  <description>
    Trust anchors (i.e. CA certs) that are used to authenticate
```

```
    client connections. Clients are authenticated if their
    certificate has a chain of trust to one of these CA
    certificates.
  </description>
  <pinned-certificate>
    <name>ca.example.com</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- trusted CA certs for random HTTPS servers on Internet -->
<pinned-certificates or:origin="or:system">
  <name>common-ca-certs</name>
  <description>
    Trusted certificates to authenticate common HTTPS servers.
    These certificates are similar to those that might be
    shipped with a web browser.
  </description>
  <pinned-certificate>
    <name>ex-certificate-authority</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- specific SSH host keys for authenticating clients -->
<pinned-host-keys or:origin="or:intended">
  <name>explicitly-trusted-ssh-host-keys</name>
  <description>
    Trusted SSH host keys used to authenticate SSH servers.
    These host keys would be analogous to those stored in
    a known_hosts file in OpenSSH.
  </description>
  <pinned-host-key>
    <name>corp-fw1</name>
    <host-key>base64encodedvalue==</host-key>
  </pinned-host-key>
</pinned-host-keys>

</trust-anchors>
```

The following example illustrates the "certificate-expiration" notification in use with the NETCONF protocol.

===== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) =====

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <trust-anchors
    xmlns="urn:ietf:params:xml:ns:yang:ietf-trust-anchors">
    <pinned-certificates>
      <name>explicitly-trusted-client-certs</name>
      <pinned-certificate>
        <name>George Jetson</name>
        <certificate-expiration>
          <expiration-date>2018-08-05T14:18:53-05:00</expiration-date>
        \e>
      </certificate-expiration>
    </pinned-certificate>
    </pinned-certificates>
  </trust-anchors>
</notification>
```

### 2.3. YANG Module

This YANG module imports modules from [RFC8341] and [I-D.ietf-netconf-crypto-types].

```
<CODE BEGINS> file "ietf-trust-anchors@2019-03-09.yang"
module ietf-trust-anchors {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-trust-anchors";
  prefix ta;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC YYYY: Common YANG Data Types for Cryptography";
  }

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

  contact
```

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

description

"This module defines a data model for configuring global trust anchors used by other data models. The data model enables the configuration of sets of trust anchors. This data model supports configuring trust anchors for both X.509 certificates and SSH host keys.

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Data Model for Global Trust Anchors";
}

/*****
/*   Typedefs for leafrefs to commonly referenced objects   */
*****/

feature x509-certificates {
  description
    "The 'x509-certificates' feature indicates that the server
    implements the /trust-anchors/pinned-certificates subtree.";
}
```

```
feature ssh-host-keys {
  description
    "The 'ssh-host-keys' feature indicates that the server
    implements the /trust-anchors/pinned-host-keys subtree.";
}

/*****
/*   Typedefs for leafrefs to commonly referenced objects   */
*****/

typedef pinned-certificates-ref {
  type leafref {
    path "/ta:trust-anchors/ta:pinned-certificates/ta:name";
    require-instance false;
  }
  description
    "This typedef enables importing modules to easily define a
    leafref to a 'pinned-certificates' object. The require
    instance attribute is false to enable the referencing of
    pinned certificates that exist only in <operational>.";
  reference
    "RFC 8342: Network Management Datastore Architecture (NMDA)";
}

typedef pinned-host-keys-ref {
  type leafref {
    path "/ta:trust-anchors/ta:pinned-host-keys/ta:name";
    require-instance false;
  }
  description
    "This typedef enables importing modules to easily define a
    leafref to a 'pinned-host-keys' object. The require
    instance attribute is false to enable the referencing of
    pinned host keys that exist only in <operational>.";
  reference
    "RFC 8342: Network Management Datastore Architecture (NMDA)";
}

/*****
/*   Protocol accessible nodes   */
*****/

container trust-anchors {
  nacm:default-deny-write;
  description
    "Contains sets of X.509 certificates and SSH host keys.";
  list pinned-certificates {
    if-feature "x509-certificates";
  }
}
```

```
key "name";
description
  "A list of pinned certificates. These certificates can be
   used by a server to authenticate clients, or by a client
   to authenticate servers. Each list of pinned certificates
   SHOULD be specific to a purpose, as the list as a whole
   may be referenced by other modules. For instance, a
   RESTCONF server's configuration might use a specific list
   of pinned certificates for when authenticating RESTCONF
   client connections.";
leaf name {
  type string;
  description
    "An arbitrary name for this list of pinned certificates.";
}
leaf description {
  type string;
  description
    "An arbitrary description for this list of pinned
    certificates.";
}
list pinned-certificate {
  key "name";
  description
    "A pinned certificate.";
  leaf name {
    type string;
    description
      "An arbitrary name for this pinned certificate. The
       name must be unique across all lists of pinned
       certificates (not just this list) so that leafrefs
       from another module can resolve to unique values.";
  }
  uses ct:trust-anchor-cert-grouping {
    refine "cert" {
      mandatory true;
    }
  }
}
}
list pinned-host-keys {
  if-feature "ssh-host-keys";
  key "name";
  description
    "A list of pinned host keys. These pinned host-keys can
     be used by clients to authenticate SSH servers. Each
     list of pinned host keys SHOULD be specific to a purpose,
     so the list as a whole may be referenced by other modules."
```

For instance, a NETCONF client's configuration might point to a specific list of pinned host keys for when authenticating specific SSH servers.";

```
leaf name {  
    type string;  
    description  
        "An arbitrary name for this list of pinned SSH  
        host keys.";  
}  
leaf description {  
    type string;  
    description  
        "An arbitrary description for this list of pinned SSH  
        host keys.";  
}  
list pinned-host-key {  
    key "name";  
    description  
        "A pinned host key."  
    leaf name {  
        type string;  
        description  
            "An arbitrary name for this pinned host-key. Must be  
            unique across all lists of pinned host-keys (not just  
            this list) so that a leafref to it from another module  
            can resolve to unique values.";  
    }  
    leaf host-key {  
        type ct:ssh-host-key;  
        mandatory true;  
        description  
            "The binary public key data for this pinned host key."  
        reference  
            "RFC YYYY: Common YANG Data Types for Cryptography";  
    }  
}  
}  
}
```

<CODE ENDS>

### 3. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.



The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/: The entire data tree defined by this module is sensitive to write operations. For instance, the addition or removal of any trust anchor may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for the entire data tree.

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments.

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

#### 4. IANA Considerations

##### 4.1. The IETF XML Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-trust-anchors  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

##### 4.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registration is requested:

name: ietf-trust-anchors  
namespace: urn:ietf:params:xml:ns:yang:ietf-trust-anchors  
prefix: ta  
reference: RFC XXXX

## 5. References

### 5.1. Normative References

- [I-D.ietf-netconf-crypto-types]  
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-02 (work in progress), October 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [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>>.
- [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>>.
- [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>>.

### 5.2. Informative References

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

- [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>>.
  
- [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>>.

## Appendix A. Change Log

## A.1. 00 to 01

- o Added features "x509-certificates" and "ssh-host-keys".
- o Added nacm:default-deny-write to "trust-anchors" container.

## A.2. 01 to 02

- o Switched "list pinned-certificate" to use the "trust-anchor-cert-grouping" from crypto-types. Effectively the same definition as before.

## A.3. 02 to 03

- o Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

## Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Martin Bjorklund, Balazs Kovacs, Eric Voit, and Liang Xia.

## Author's Address

Kent Watsen  
Watsen Networks

EMail: kent+ietf@watsen.net

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
March 9, 2019

YANG Groupings for HTTP Clients and HTTP Servers  
draft-kwatsen-netconf-http-client-server-00

Abstract

This document defines two YANG modules: the first defines a grouping for configuring a generic HTTP client, and the second defines a grouping for configuring a generic HTTP server. It is intended that these groupings will be used by applications using the HTTP protocol.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

Status of This Memo

This Internet-Draft 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 September 10, 2019.

## Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. The HTTP Client Model . . . . .	3
3.1. Tree Diagram . . . . .	3
3.2. Example Usage . . . . .	4
3.3. YANG Module . . . . .	5
4. The HTTP Server Model . . . . .	9
4.1. Tree Diagram . . . . .	9
4.2. Example Usage . . . . .	10
4.3. YANG Module . . . . .	10
5. Security Considerations . . . . .	12
6. IANA Considerations . . . . .	13
6.1. The IETF XML Registry . . . . .	13
6.2. The YANG Module Names Registry . . . . .	13
7. References . . . . .	14
7.1. Normative References . . . . .	14
7.2. Informative References . . . . .	14
Author's Address . . . . .	15

## 1. Introduction

This document defines two YANG 1.1 [RFC7950] modules: the first defines a grouping for configuring a generic HTTP client, and the second defines a grouping for configuring a generic HTTP server. It is intended that these groupings will be used by applications using the HTTP protocol. For instance, these groupings could help define the configuration module for an SSH, TLS, or HTTP based application.

## 2. Terminology

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

## 3. The HTTP Client Model

### 3.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-http-client" module.

module: ietf-http-client

```
grouping http-client-grouping
+-- http-client-identity
|   +-- (auth-type)?
|   |   +--:(basic)
|   |   |   +-- basic
|   |   |   |   +-- user-id?    string
|   |   |   |   +-- password?  string
|   |   +--:(bearer)
|   |   |   +-- bearer
|   |   |   |   +-- token?    string
|   |   +--:(digest)
|   |   |   +-- digest
|   |   |   |   +-- username?  string
|   |   |   |   +-- password?  string
|   |   +--:(hoba)
|   |   |   +-- hoba
|   |   +--:(mutual)
|   |   |   +-- mutual
|   |   +--:(negotiate)
|   |   |   +-- negotiate
|   |   +--:(oauth)
|   |   |   +-- oauth
|   |   +--:(scram-sha-1)
|   |   |   +-- scram-sha-1
|   |   +--:(scram-sha-256)
|   |   |   +-- scram-sha-256
|   |   +--:(vapid)
|   |   |   +-- vapid
+-- http-keepalives {http-client-keepalives}?
|   +-- max-wait?        uint16
|   +-- max-attempts?    uint8
```

```
grouping http-client-identity-grouping
  +-- http-client-identity
    +-- (auth-type)?
      +--:(basic)
        |   +-- basic
        |     +-- user-id?    string
        |     +-- password?   string
      +--:(bearer)
        |   +-- bearer
        |     +-- token?      string
      +--:(digest)
        |   +-- digest
        |     +-- username?    string
        |     +-- password?    string
      +--:(hoba)
        |   +-- hoba
      +--:(mutual)
        |   +-- mutual
      +--:(negotiate)
        |   +-- negotiate
      +--:(oauth)
        |   +-- oauth
      +--:(scram-sha-1)
        |   +-- scram-sha-1
      +--:(scram-sha-256)
        |   +-- scram-sha-256
      +--:(vapid)
        |   +-- vapid
grouping http-keepalives-grouping
  +-- http-keepalives {http-client-keepalives}?
    +-- max-wait?      uint16
    +-- max-attempts?  uint8
```

### 3.2. Example Usage

This section presents an example showing the http-client-grouping populated with some data.



```
<http-client xmlns="urn:ietf:params:xml:ns:yang:ietf-http-client">
  <http-client-identity>
    <basic>
      <user-id>bob</user-id>
      <password>secret</password>
    </basic>
  </http-client-identity>
  <http-keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </http-keepalives>
</http-client>
```

### 3.3. YANG Module

This YANG module has normative references to [RFC6991].

```
<CODE BEGINS> file "ietf-http-client@2019-03-09.yang"
module ietf-http-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-http-client";
  prefix httpc;

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";
  description
    "This module defines reusable groupings for HTTP clients that
    can be used as a basis for specific HTTP client instances.

    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.

    Copyright (c) 2019 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD
    License set forth in Section 4.c of the IETF Trust's
    Legal Provisions Relating to IETF Documents
```

```
(http://trustee.ietf.org/license-info).

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

revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for HTTP Clients and HTTP Servers";
}

// Features

feature http-client-keepalives {
  description
    "Per socket HTTP keepalive parameters are configurable for
    HTTP clients on the server implementing this feature.";
}

// Groupings

grouping http-client-grouping {
  description
    "A reusable grouping for configuring a HTTP client,
    including the IP address and port number it initiates
    a connections to.";
  uses http-client-identity-grouping;
  uses http-keepalives-grouping;
  // uses http-proxy-grouping;
}

grouping http-client-identity-grouping {
  description
    "A reusable grouping for configuring a HTTP client identity.";
  container http-client-identity {
    description
      "The credentials used by the client to authenticate to
      the HTTP server.";
    choice auth-type {
      description
        "The authentication type.";
      container basic {
        leaf user-id {
          type string;
          description
            "The user-id for the authenticating client.";
        }
      }
    }
  }
}
```

```
    leaf password {
      type string;
      description
        "The password for the authenticating client.";
    }
    description
      "The 'basic' HTTP scheme credentials.";
    reference
      "RFC 7617: The 'Basic' HTTP Authentication Scheme";
  }
  container bearer {
    leaf token {
      type string;
      description
        "The bearer token for the authenticating client,
        encoded in base64, as described in RFC 6750,
        Section 2.1.";
    }
    description
      "The 'bearer' HTTP scheme credentials.";
    reference
      "RFC 6750: The OAuth 2.0 Authorization Framework:
      Bearer Token Usage";
  }
  container digest {
    leaf username {
      type string;
      description
        "The username for the authenticating client.";
    }
    leaf password {
      type string;
      description
        "The password for the authenticating client.";
    }
    description
      "The 'digest' HTTP scheme credentials.";
    reference
      "RFC 7616: HTTP Digest Access Authentication";
  }
  container hoba {
    // FIXME
    description
      "The 'hoba' HTTP scheme credentials.";
    reference
      "RFC 7486: HTTP Origin-Bound Authentication (HOBA)";
  }
  container mutual {
```

```
    // FIXME
    description
        "The 'mutual' HTTP scheme credentials.";
    reference
        "RFC 8120: Mutual Authentication Protocol for HTTP";
}
container negotiate {
    // FIXME
    description
        "The 'negotiate' HTTP scheme credentials.";
    reference
        "RFC 4559: SPNEGO-based Kerberos and NTLM HTTP
        Authentication in Microsoft Windows";
}
container oauth {
    // FIXME
    description
        "The 'oauth' HTTP scheme credentials.";
    reference
        "RFC 6749: The OAuth 2.0 Authorization Framework";
}
container scram-sha-1 {
    // FIXME
    description
        "The 'scram-sha-1' HTTP scheme credentials.";
    reference
        "RFC 7804: Salted Challenge Response HTTP
        Authentication Mechanism";
}
container scram-sha-256 {
    // FIXME
    description
        "The 'scram-sha-256' HTTP scheme credentials.";
    reference
        "RFC 7804: Salted Challenge Response HTTP
        Authentication Mechanism";
}
container vapid {
    // FIXME
    description
        "The 'vapid' HTTP scheme credentials.";
    reference
        "RFC 8292: Voluntary Application Server
        Identification (VAPID) for Web Push";
}
}
}
```

```
grouping http-keepalives-grouping {
  description
    "A reusable grouping for configuring HTTP client keepalive
    parameters.";
  container http-keepalives {
    if-feature "http-client-keepalives";
    description
      "Configures the keep-alive policy, to proactively test
      the aliveness of the HTTP server. An unresponsive
      HTTP server is dropped after approximately max-wait
      * max-attempts seconds.";
    leaf max-wait {
      type uint16 {
        range "1..max";
      }
      units "seconds";
      default "30";
      description
        "Sets the amount of time in seconds after which if no
        data has been received from the HTTP server, a HTTP
        level message will be sent to test the aliveness of
        the HTTP server.";
    }
    leaf max-attempts {
      type uint8;
      default "3";
      description
        "Sets the maximum number of sequential keep-alive messages
        that can fail to obtain a response from the HTTP server
        before assuming the HTTP server is no longer alive.";
    }
  }
}
}
}
}
<CODE ENDS>
```

#### 4. The HTTP Server Model

##### 4.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-http-server" module.

```
module: ietf-http-server

  grouping http-server-grouping
    +-- http-keepalives {http-server-keepalives}?
      +-- max-wait?      uint16
      +-- max-attempts?  uint8
  grouping keepalives-grouping
    +-- http-keepalives {http-server-keepalives}?
      +-- max-wait?      uint16
      +-- max-attempts?  uint8
```

#### 4.2. Example Usage

This section presents an example showing the http-server-grouping populated with some data.

```
<http-server xmlns="urn:ietf:params:xml:ns:yang:ietf-http-server">
  <http-keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </http-keepalives>
</http-server>
```

#### 4.3. YANG Module

This YANG module has normative references to [RFC6991].

```
<CODE BEGINS> file "ietf-http-server@2019-03-09.yang"
module ietf-http-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-http-server";
  prefix https;

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";
  description
    "This module defines reusable groupings for HTTP servers that
    can be used as a basis for specific HTTP server instances.

    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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for HTTP Clients and HTTP Servers";
}

// Features

feature http-server-keepalives {
  description
    "Per socket HTTP keepalive parameters are configurable for
    HTTP servers on the server implementing this feature.";
}

// Groupings

grouping http-server-grouping {
  description
    "A reusable grouping for configuring a HTTP server,
    including the IP address and port number it listens
    for connections on.";
  uses keepalives-grouping;
}

grouping keepalives-grouping {
  description
    "A reusable grouping for configuring HTTP server keepalive
    parameters.";
  container http-keepalives {
    if-feature "http-server-keepalives";
    description
      "Configures the keep-alive policy, to proactively test
```

```
        the aliveness of the HTTP client.  An unresponsive
        HTTP client is dropped after approximately max-wait
        * max-attempts seconds.";
    leaf max-wait {
        type uint16 {
            range "1..max";
        }
        units "seconds";
        default "30";
        description
            "Sets the amount of time in seconds after which if no
            data has been received from the HTTP client, a HTTP
            level message will be sent to test the aliveness of
            the HTTP client.";
    }
    leaf max-attempts {
        type uint8;
        default "3";
        description
            "Sets the maximum number of sequential keep-alive messages
            that can fail to obtain a response from the HTTP client
            before assuming the HTTP client is no longer alive.";
    }
}
}
}
}
<CODE ENDS>
```

## 5. Security Considerations

The YANG modules defined in this document are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, HTTP) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the modules defined in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

There are a number of data nodes defined in the YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config)



to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

NONE

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

NONE

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

NONE

## 6. IANA Considerations

### 6.1. 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-http-client  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-http-server  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

### 6.2. 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-http-client
namespace: urn:ietf:params:xml:ns:yang:ietf-http-client
prefix:    httpc
reference:  RFC XXXX

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

## 7. References

### 7.1. Normative References

- [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>>.
- [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>>.
- [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>>.
- [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>>.

### 7.2. Informative References

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

- [RFC6241]    Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,  
              and A. Bierman, Ed., "Network Configuration Protocol  
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,  
              <<https://www.rfc-editor.org/info/rfc6241>>.
  
- [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>>.
  
- [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>>.

Author's Address

Kent Watsen  
Watsen Networks

EMail: [kent+ietf@watsen.net](mailto:kent+ietf@watsen.net)

NETCONF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

K. Watsen  
Watsen Networks  
March 9, 2019

YANG Groupings for TCP Clients and TCP Servers  
draft-kwatsen-netconf-tcp-client-server-00

Abstract

This document defines two YANG modules: the first defines a grouping for configuring a generic TCP client, and the second defines a grouping for configuring a generic TCP server. It is intended that these groupings will be used by applications using the TCP protocol.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

- o "2019-03-09" --> the publication date of this draft

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

- o Appendix A. Change Log

Status of This Memo

This Internet-Draft 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 September 10, 2019.

## Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. The TCP Client Model . . . . .	3
3.1. Tree Diagram . . . . .	3
3.2. Example Usage . . . . .	3
3.3. YANG Module . . . . .	4
4. The TCP Server Model . . . . .	7
4.1. Tree Diagram . . . . .	7
4.2. Example Usage . . . . .	8
4.3. YANG Module . . . . .	8
5. Security Considerations . . . . .	11
6. IANA Considerations . . . . .	12
6.1. The IETF XML Registry . . . . .	12
6.2. The YANG Module Names Registry . . . . .	13
7. References . . . . .	13
7.1. Normative References . . . . .	13
7.2. Informative References . . . . .	14
Author's Address . . . . .	14

## 1. Introduction

This document defines two YANG 1.1 [RFC7950] modules: the first defines a grouping for configuring a generic TCP client, and the second defines a grouping for configuring a generic TCP server. It is intended that these groupings will be used by applications using the TCP protocol. For instance, these groupings could help define the configuration module for an SSH, TLS, or HTTP based application.

## 2. Terminology

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

## 3. The TCP Client Model

### 3.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-tcp-client" module.

```
module: ietf-tcp-client

  grouping tcp-client-grouping
    +-- remote-address      inet:host
    +-- remote-port?       inet:port-number
    +-- local-address?     inet:ip-address
    +-- local-port?        inet:port-number
    +-- tcp-keepalives {tcp-client-keepalives}?
      +-- idle-time?       uint16
      +-- max-probes?      uint16
      +-- probe-interval?  uint16
  grouping ip-params-grouping
    +-- remote-address      inet:host
    +-- remote-port?       inet:port-number
    +-- local-address?     inet:ip-address
    +-- local-port?        inet:port-number
  grouping keepalives-grouping
    +-- tcp-keepalives {tcp-client-keepalives}?
      +-- idle-time?       uint16
      +-- max-probes?      uint16
      +-- probe-interval?  uint16
```

### 3.2. Example Usage

This section presents an example showing the tcp-client-grouping populated with some data.

```
<tcp-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-client">
  <remote-address>www.example.com</remote-address>
  <remote-port>443</remote-port>
  <local-address>0.0.0.0</local-address>
  <local-port>0</local-port>
  <tcp-keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </tcp-keepalives>
</tcp-client>
```

### 3.3. YANG Module

This YANG module has normative references to [RFC6991].

```
<CODE BEGINS> file "ietf-tcp-client@2019-03-09.yang"
module ietf-tcp-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-client";
  prefix tcpc;

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

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

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

  description
    "This module defines reusable groupings for TCP clients that
    can be used as a basis for specific TCP client instances.

    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.

    Copyright (c) 2019 IETF Trust and the persons identified as
```

authors of the code. All rights reserved.

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

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

```
revision 2019-03-09 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for TCP Clients and TCP Servers";
}

// Features

feature tcp-client-keepalives {
  description
    "Per socket TCP keepalive parameters are configurable for
    TCP clients on the server implementing this feature.";
}

// Groupings

grouping tcp-client-grouping {
  description
    "A reusable grouping for configuring a TCP client.";
  uses ip-params-grouping;
  uses keepalives-grouping;
}

grouping ip-params-grouping {
  description
    "A reusable grouping for configuring TCP client IP level
    parameters.";
  leaf remote-address {
    type inet:host;
    mandatory true;
    description
      "The IP address or hostname of the remote peer to connect to.
      If a domain name is configured, then the DNS resolution
      should happen on each connection attempt. If the the DNS
      resolution results in multiple IP addresses, the IP addresses
```



```
        are tried according to local preference order until a
        connection has been established or until all IP addresses
        have failed.";
    }
    leaf remote-port {
        type inet:port-number;
        default "0";
        description
            "The IP port number for the remote peer to connect to. An
            invalid default value (0) is used (instead of 'mandatory
            true') so that a application level data model may 'refine'
            it with an application specific default port number value.";
    }
    leaf local-address {
        type inet:ip-address;
        description
            "The local IP address/interface (VRF?) to bind to for when
            connecting to the remote peer. INADDR_ANY ('0.0.0.0') or
            INADDR6_ANY ('0:0:0:0:0:0:0:0' a.k.a. '::') MAY be used to
            explicitly indicate the implicit default, that the server
            can bind to any IPv4 or IPv6 addresses, respectively.";
    }
    leaf local-port {
        type inet:port-number;
        default "0";
        description
            "The local IP port number to bind to for when connecting to
            the remote peer. The port number '0', which is the default
            value, indicates that any available local port number may
            be used.";
    }
}

grouping keepalives-grouping {
    description
        "A reusable grouping for configuring TCP client keepalive
        parameters.";
    container tcp-keepalives {
        if-feature "tcp-client-keepalives";
        description
            "Configures the keep-alive policy, to proactively test the
            aliveness of the TCP server. Not all clients will use
            all the values, based on capabilities of the underlying
            operating system. An unresponsive TCP server is dropped
            after approximately (idle-time * 60) + (max-probes *
            probe-interval) seconds.";
        leaf idle-time {
            type uint16 {
```

```
        range "1..max";
    }
    units "minutes";
    description
        "Sets the amount of time in minutes after which if no data
        has been received from the TCP server, a TCP-level probe
        message will be sent to test the aliveness of the TCP
        server. When 'idle-time' is not configured (the default)
        TCP keep-alives are disabled.";
    }
    leaf max-probes {
        type uint16 {
            range "1..max";
        }
        description
            "Sets the maximum number of sequential keep-alive probes
            that can fail to obtain a response from the TCP server
            before assuming the TCP server is no longer alive. If
            no value is specified, then the operating system provided
            default value is used.";
    }
    leaf probe-interval {
        type uint16 {
            range "1..max";
        }
        units "seconds";
        description
            "Sets the time interval between failed probes. If no value
            is specified, then the operating system provided default
            value is used.";
    }
    }
}
}
}
<CODE ENDS>
```

#### 4. The TCP Server Model

##### 4.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-tcp-server" module.

```
module: ietf-tcp-server

  grouping tcp-server-grouping
    +-- local-address      inet:ip-address
    +-- local-port?        inet:port-number
    +-- tcp-keepalives {tcp-server-keepalives}?
      +-- idle-time?       uint16
      +-- max-probes?       uint16
      +-- probe-interval?  uint16
  grouping ip-params-grouping
    +-- local-address      inet:ip-address
    +-- local-port?        inet:port-number
  grouping keepalives-grouping
    +-- tcp-keepalives {tcp-server-keepalives}?
      +-- idle-time?       uint16
      +-- max-probes?       uint16
      +-- probe-interval?  uint16
```

#### 4.2. Example Usage

This section presents an example showing the tcp-server-grouping populated with some data.

```
<tcp-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tcp-server">
  <local-address>10.20.30.40</local-address>
  <local-port>7777</local-port>
  <tcp-keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </tcp-keepalives>
</tcp-server>
```

#### 4.3. YANG Module

This YANG module has normative references to [RFC6991].

```
<CODE BEGINS> file "ietf-tcp-server@2019-03-09.yang"
module ietf-tcp-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tcp-server";
  prefix tcps;

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

## organization

"IETF NETCONF (Network Configuration) Working Group";

## contact

"WG Web: <<http://datatracker.ietf.org/wg/netconf/>>

WG List: <<mailto:netconf@ietf.org>>

Author: Kent Watsen <<mailto:kent+ietf@watsen.net>>;

## description

"This module defines reusable groupings for TCP servers that can be used as a basis for specific TCP server instances.

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.

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

## revision 2019-03-09 {

## description

"Initial version";

## reference

"RFC XXXX: YANG Groupings for TCP Clients and TCP Servers";

}

## // Features

## feature tcp-server-keepalives {

## description

"Per socket TCP keepalive parameters are configurable for TCP servers on the server implementing this feature."

}

## // Groupings

```
grouping tcp-server-grouping {
  description
    "A reusable grouping for configuring a TCP server.";
  uses ip-params-grouping;
  uses keepalives-grouping;
}

grouping ip-params-grouping {
  description
    "A reusable grouping for configuring TCP server IP level
    parameters.";
  leaf local-address {
    type inet:ip-address;
    mandatory true;
    description
      "The local IP address to listen on for incoming TCL
      client connections.  INADDR_ANY (0.0.0.0) or INADDR6_ANY
      (0:0:0:0:0:0:0:0 a.k.a. ::) MUST be used when the server
      is to listen on all IPv4 or IPv6 addresses, respectively.";
  }
  leaf local-port {
    type inet:port-number;
    default "0";
    description
      "The local port number to listen on for incoming TCP client
      connections.  An invalid default value (0) is used (instead
      of 'mandatory true') so that a application level data model
      may 'refine' it with an application specific default port
      number value.";
  }
}

grouping keepalives-grouping {
  description
    "A reusable grouping for configuring TCP server keepalive
    parameters.";
  container tcp-keepalives {
    if-feature "tcp-server-keepalives";
    description
      "Configures the keep-alive policy, to proactively test the
      aliveness of the TCP client.  Not all servers will use
      all the values, based on capabilities of the underlying
      operating system.  An unresponsive TCP client is dropped
      after approximately (idle-time * 60) + (max-probes *
      probe-interval) seconds.";
    leaf idle-time {
      type uint16 {
        range "1..max";
      }
    }
  }
}
```

```

    }
    units "minutes";
    description
        "Sets the amount of time in minutes after which if no data
        has been received from the TCP client, a TCP-level probe
        message will be sent to test the aliveness of the TCP
        client. When 'idle-time' is not configured (the default)
        TCP keep-alives are disabled.";
}
leaf max-probes {
    type uint16 {
        range "1..max";
    }
    description
        "Sets the maximum number of sequential keep-alive probes
        that can fail to obtain a response from the TCP client
        before assuming the TCP client is no longer alive. If
        no value is specified, then the operating system provided
        default value is used.";
}
leaf probe-interval {
    type uint16 {
        range "1..max";
    }
    units "seconds";
    description
        "Sets the time interval between failed probes. If no value
        is specified, then the operating system provided default
        value is used.";
}
}
}
}
<CODE ENDS>

```

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Since the modules defined in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

There are a number of data nodes defined in the YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

NONE

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

NONE

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

NONE

## 6. IANA Considerations

### 6.1. 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-tcp-client  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tcp-server  
Registrant Contact: The NETCONF WG of the IETF.  
XML: N/A, the requested URI is an XML namespace.

## 6.2. 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-tcp-client
namespace: urn:ietf:params:xml:ns:yang:ietf-tcp-client
prefix:    tcpc
reference:  RFC XXXX

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

## 7. References

### 7.1. Normative References

- [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>>.
- [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>>.
- [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>>.
- [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>>.



## 7.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [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>>.
- [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>>.

## Author's Address

Kent Watsen  
Watsen Networks

EMail: [kent+ietf@watsen.net](mailto:kent+ietf@watsen.net)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 11, 2019

R. Wilton  
Cisco Systems, Inc.  
March 10, 2019

YANG Packages  
draft-rwilton-netmod-yang-packages-01

Abstract

This document defines YANG packages, an organizational structure holding a set of related YANG modules, that can be used to simplify the conformance and sharing of YANG schema. It describes how YANG instance data documents are used to define YANG packages, and how the YANG library information published by a server can be augmented with additional packaging related information.

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 September 11, 2019.

Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Terminology and Conventions . . . . .	2
2. Introduction . . . . .	3
3. Background on YANG packaging . . . . .	4
4. Possible alternative solutions . . . . .	4
4.1. Using module tags . . . . .	4
4.2. Using YANG library . . . . .	5
5. Objectives . . . . .	6
6. Package description . . . . .	7
6.1. Package definition rules . . . . .	7
6.2. Package versioning . . . . .	8
6.3. Client server package conformance . . . . .	10
6.4. Schema referential completeness . . . . .	10
6.5. Submodules packaging considerations . . . . .	11
6.6. Revision history . . . . .	11
6.7. Uniqueness of packages and global registry . . . . .	11
7. YANG Packaging instance data . . . . .	12
8. YANG Packaging additions to YANG library . . . . .	13
8.1. Package List . . . . .	14
8.2. Binding from schema to package . . . . .	14
8.3. Tree diagram . . . . .	15
9. YANG Packaging groupings . . . . .	15
10. YANG Modules . . . . .	17
11. Security Considerations . . . . .	29
12. IANA Considerations . . . . .	30
13. Open Questions/Issues . . . . .	31
14. Acknowledgements . . . . .	31
15. References . . . . .	31
15.1. Normative References . . . . .	31
15.2. Informative References . . . . .	32
Appendix A. Tree output for ietf-yang-library with package augmentations . . . . .	32
Appendix B. Examples . . . . .	34
B.1. Example IETF Network Device YANG package . . . . .	35
B.2. Example IETF Basic Routing YANG package . . . . .	37
B.3. Package import conflict resolution example . . . . .	40
Author's Address . . . . .	43

## 1. Terminology and Conventions

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.

This document uses terminology introduced in the YANG versioning requirements draft [I-D.verdt-netmod-yang-versioning-reqs].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- o datastore schema

In addition, this document makes use of the following terminology:

- o bc: Used as an abbreviation for a backwards-compatible change.
- o nbc: Used as an abbreviation for a non-backwards-compatible change.
- o editorial change: A backwards-compatible change that does not change the YANG module semantics in any way.

Note - the bc/nbc/editorial terminology should probably be defined and referenced from the YANG module versioning solution draft.

## 2. Introduction

This document defines and describes the YANG [RFC7950] constructs that are used to define and use YANG packages.

A YANG package is an organizational structure that groups a set of related YANG modules together into a consistent versioned definition. YANG packages can themselves refer to and reuse other package definitions.

The draft consists of the following significant sections:

A background section that describes some of the prior work in this area, both within IETF and the wider industry.

An overview of the objectives for a YANG packaging solution, and also what work is out of scope for this document.

The definition of YANG packages, how package definitions are constructed, and how they are used.

How YANG instance data documents [I-D.ietf-netmod-yang-instance-file-format] are used to define particular YANG package instances.

Augmentations to the YANG library [I-D.ietf-netconf-rfc7895bis] content published by servers to include YANG packaging related information.

YANG modules the provide the definitions for YANG packages.

Non-normative examples of YANG package instances are provided in the appendicies.

### 3. Background on YANG packaging

It has long been acknowledged within the IETF NETMOD community that network management using YANG requires a unit of organization and conformance that is broader in scope than individual YANG modules.

'The YANG Package Statement' [I-D.bierman-netmod-yang-package] proposed a YANG package mechanism based on new YANG language statements, where a YANG package is defined in a file similar to how YANG modules are defined, and would require enhancements to YANG compilers to understand the new statements used to define particular package instances. This document did not progress in the working group, although this may have been due to other higher priority concerns or resource constraints within the working group rather than due to consideration of the technical merits of the proposed approach.

OpenConfig [openconfigsemver] describes an approach to versioning 'bundle releases' based on git tags. I.e. a set of modules, at particular versions, can be marked with the same release tag to indicate that they are known to interoperate together.

The NETMOD WG in general, and the YANG versioning design team in particular, are exploring solutions to the YANG versioning requirements, [I-D.verdt-netmod-yang-versioning-reqs]. Solutions to the versioning requirements can be split into several distinct areas. One draft, TBD (draft-verdt-netmod-yang-semver), has a primary focus on YANG versioning scoped to individual modules. But an overall solution should also consider YANG versioning and conformance scoped to a server's datastore schema. YANG packages may help form part of the solution for versioning at the datastore schema level.

### 4. Possible alternative solutions

#### 4.1. Using module tags

Module tags have been suggested as an alternative solution, and indeed that can address some of the same requirements as YANG packages but not all of them.

Module tags can be used to group or organize YANG modules. However, this raises the question of where this tag information is stored. Module tags either require that the YANG module files themselves are updated with the module tag information (creating another versioning problem), or for the module tag information to be hosted elsewhere, perhaps in a centralized YANG Catalog, or in instance data documents similar to how YANG packages have been defined in this draft.

One of the principle aims of YANG packages is to be a versioned object that defines a precise set of YANG modules versions that work together. Module tags cannot meet this aim without an explosion of module tags definitions (i.e. a separate module tag must be defined for each package version).

Module tags cannot support the hierarchical scheme to construct YANG schema that is proposed in this draft.

#### 4.2. Using YANG library

Another question is whether it is necessary to define new YANG modules to define YANG packages, and whether YANG library could just be reused in an instance data document. The use of YANG packages is intended to offer several benefits over just using YANG library:

1. Packages allow schema to be built in a hierarchical fashion. [I-D.ietf-netconf-rfc7895bis] only allows one layer of hierarchy (using module sets), and there can be no conflicts between module revisions in different module-sets.
2. Packages can be made available off the box, with a well defined unique name, avoiding the need for clients to download, and construct/check the entire YANG schema for each device, instead they can rely on the named packages. YANG libraries use of checksums are unique only to the device that generated them.
3. Packages are versioned using a semantic versioning scheme, YANG library does not have a schema level semantic version number, although this could potentially be added if required.
4. For a YANG library instance data document to contain the necessary information, it needs both YANG library, and probably also various augmentations (e.g. to include each module's semantic version number), unless a new version of YANG library is defined containing this information. A module definition for a YANG package could be defined to contain all of the necessary information to solve the problem.

5. YANG library is designed to publish information about the modules, datastores, and datastore schema used by a server. It is unclear whether exactly the same information is required for an offline schema definition, or whether these definitions might deviate from each other over time. E.g., some thought needs to be given concerning the relationship between datastores and schema.

## 5. Objectives

The main goals of YANG package definitions include, but are not restricted to:

- o To act as a simplified YANG conformance mechanism. Rather than conformance being performed against a set of individual YANG module revisions, conformance could also be more simply stated in terms of YANG packages, with a set modifications (e.g. additional modules, deviations, or features).
- o To allow YANG datastore schema to be specified in a more concise way rather than having to list all modules and revisions. YANG package definitions can be defined in documents that can be referenced by a URL rather than requiring explicit lists of modules to be shared between client and server. Hence, a YANG package must contain sufficient information to allow a client or server to precisely construct the schema associated with the package.
- o To provide generic packaging related YANG grouping definitions for use in other YANG modules, as required.
- o To define a mainly linear versioned history of sets of modules versions that are known to work together. I.e. to help mitigate the problem were a client must manage devices from multiple vendors, and vendor A implements version 1.0.0 of module foo and version 2.0.0 of module bar, and vendor B implements version 2.0.0 of module foo and version 1.0.0 of module bar. For a client, trying to interoperate with multiple vendors, and many YANG modules, then finding a consistent lowest common denominator set of YANG module versions may be difficult, or impossible.

Protocol mechanisms of how clients could negotiate which packages or package versions are be used for client server communications are outside the scope of this document. However, the design of the YANG library augmentations for YANG packages are intended to keep open the possibility of such extensions in future work.

Finally, the package definitions proposed by this document are intended to be relatively basic in their definition and the functionality that they support. As industry gains experience using YANG packages, the standard YANG mechanisms of updating, or augmenting, YANG modules could be used to extend the functionality supported by YANG packages.

## 6. Package description

This document specifies an approach to defining YANG packages that is different to either of the approaches described in the background.

The approach defined here is for a YANG package definition structure to be defined using existing YANG language statements without requiring extensions or new YANG statements. By making use of this structure, particular YANG package instances can be defined as YANG instance data documents [I-D.ietf-netmod-yang-instance-file-format] with well defined names and locations.

The YANG semantic versioning scheme, described in draft-verdt-netmod-yang-semver (TBD), is used to version YANG packages using an equivalent scheme to how individual YANG modules version numbers are changed.

YANG library is augmented to allow servers to report the packages that they implement and to associate those packages back to particular datastore schema.

TODO - It would be helpful if the YANG instance data file format [I-D.ietf-netmod-yang-instance-file-format] could also reference a YANG packages to specify the schema associated with an instance data document. This could either be defined in instance-file-format draft, or as a YANG augmentation as part of this draft.

Each version of a YANG package defines: a set of YANG modules that are implemented at particular versions or revisions; a set of YANG modules that are import-only with particular versions or revisions; and a set of mandatory module features that implementations of the package MUST implement or otherwise deviate.

### 6.1. Package definition rules

The following rules define how packages are defined:

Every YANG package definition MUST be referentially complete. I.e. all import and include statements for all YANG modules included in a package MUST resolve to a module specified in the package itself, or an imported package.



For a given package, each separate instance of the package MUST have a unique version number that follows the semantic versioning rules described in Section 6.2.

A package MAY have a revision-date. Any package revision-dates MUST be unique for different package versions.

For each module implemented by a package, only a single revision/version MUST be implemented.

The version/revision of a module listed in the package module list supercedes any version/revision of the module listed in a imported package module list. This allows a package to resolve any conflicting implemented module versions/revisions in imported packages.

The replaces-revision leaf-list in the import-only-module list can be used to exclude duplicate revisions of import-only modules from imported packages. Otherwise, the import-only-modules for a package are the import-only-modules from all imported packages combined with any modules listed in the packages import-only-module list.

Modules referenced by a package SHOULD specify the version of the module, both in the package definition and within the module definition itself.

Modules referenced by a package MUST specify the revision date of the module, both in the package definition and within the module definition itself.

## 6.2. Package versioning

Every YANG package must specify a YANG semantic version field that defines the particular version of the package.

The rules for incrementing the YANG package version number are equivalent to the semantic versioning rules used to version individual YANG modules, defined in TBD (draft-verdt-netmod-yang-semver).

The semantic versioning rules, as they apply to YANG packages, are defined using the following two step process:

The first step is to determined whether the change to the YANG package is classified as a major, minor, or editorial based on the content that has changed in the package relative to the previous version. Where available, the semantic version number of the

referenced elements in the package (imported packages or modules) can be used to help determine what type of change is being made. The formal rules are:

If any of the referenced elements of the package (imported packages or modules) are changed in an nbc way, or if any imported package, module, or mandatory-feature is removed from the package definition, then the package has been updated in an nbc way.

If none of the referenced elements of the package (imported packages, modules) are removed or changed in a nbc way, but some referenced elements are changed in a bc way, or new referenced elements or mandatory-features added, then the package is deemed to be updated in a bc way.

If none of the referenced elements of the package (imported packages, modules) are added, removed, or changed in a nbc or bc way, but some referenced elements have editorial changes then the package is deemed to be updated in an editorial way.

The second step, after it has been determined what type of version change is being made to the YANG package, is for the YANG semantic versioning rules to be applied to update the YANG package semantic version number. The formal rules are:

If the package is being updated in a nbc way, then the package version "X.Y.Z[m|M]" SHOULD be updated to "X+1.0.0" unless that package version has already been defined with different content, in which case the package version "X.Y.Z+1M MUST be used instead.

If the package is being updated in a bc way, then the package version "X.Y.Z[m|M]" SHOULD be updated to "X.Y+1.0" unless that package version has already been defined with different content, in which case if the current package version is "X.Y.ZM" then it MUST be updated to "X.Y.Z+1M", or otherwise "X.Y.Z+1m".

If the package is being updated in an editorial way, then the package version "X.Y.Z[m|M]" MUST be updated to "X.Y.Z+1[m|M]", retaining the 'm|M' character if it is already present in the previous version."

Package YANG semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta definitions and need not follow the nbc rules, and the minor version number can be incremented instead.

In all cases, the 3 number fields that comprise a YANG semantic version number associated with a YANG package MUST uniquely identify the contents of that YANG package.

### 6.3. Client server package conformance

The YANG semantic versioning scheme used for YANG packages means that a client can determine the nature of changes between two package revisions.

This means that a client is not restricted to working only with servers that advertise exactly the same version of package in YANG library. Instead, reasonable clients should be able to interoperate with a server that supports a package version that is backwards compatible to what the client is designed for.

For example, a client coded to support 'foo' package at version 1.0.0 should interoperate with a server implementing 'foo' package at version 1.3.5, because the YANG semantic versioning rules require that package version 1.3.5 is backwards compatible to version 1.0.0.

This also has a relevance on servers that are capable of supporting version selection because they need not necessarily support every version of a YANG package to ensure good client compatibility. Choosing suitable minor versions within each major version number should generally be sufficient, particular if they can avoid NBC patch level changes (i.e. 'M' labelled versions).

### 6.4. Schema referential completeness

A YANG package may represent a schema that is 'referentially complete', or 'referentially incomplete'.

If all import statements in all YANG modules included in the package (either directly, or through imported packages) can be resolved to a module revision defined with the YANG package definition, then the package is classified as referentially complete. Conversely, if one or more import statements cannot be resolved to a module specified as part of the package definition, then the package is classified as referentially incomplete.

A package that represents the exact contents of a datastore schema MUST always be referentially complete.

Referentially incomplete packages can be used to group sets of logically related modules together, but without requiring a fixed dependency on all imported 'types' modules, instead leaving the choice of specific revisions of 'types' modules to be resolved when the package definition is used.

### 6.5. Submodules packaging considerations

As defined in [RFC7950] and draft-verdt-netmod-yang-semver (TBD), YANG conformance and versioning is specified in terms of particular revisions of YANG modules rather than for individual submodules.

However, YANG package definitions also include the list of submodules included by a module, primarily to provide a location of where the submodule definition can be obtained from, allowing a YANG schema to be fully constructed from a YANG package instance-data definition.

Restructuring how a module uses, or does not use, submodules is treated as an editorial level change in YANG semantic versioning, on the condition that there is no change in the modules semantic behavior due to the restructuring.

To ensure that a module and any constituent submodule are tightly related, all 'include' statements in a YANG module SHOULD specify revision-dates of the included submodules. If 'include' statement revision-dates are included in the YANG module then they MUST match the 'revision' field specified for the submodule in the packages's submodules lists.

### 6.6. Revision history

YANG packages do not contain a revision history, because a linear revision history does not work well for a versioning object that supports branching. In addition, some packages could have frequent revisions, and a long revision history would bloat the package definition.

To mitigate this, the package definition includes a 'previous-version' leaf that indicates the specific version this package definition is based on. By recursively examining the 'previous-version' leaf of a package definition, a full revision history can be dynamically constructed if required.

### 6.7. Uniqueness of packages and global registry

The name given to a package SHOULD be globally unique, and it SHOULD include an appropriate organization prefix in the name, equivalent to YANG module naming conventions.

Ideally a YANG instance data document defining a particular package version would be publically available at one or more URLs.

## 7. YANG Packaging instance data

YANG packages are expected to be defined as YANG instance data documents [I-D.ietf-netmod-yang-instance-file-format] using the YANG schema below to define the package data itself.

The instance data document for each version of a YANG package SHOULD be made available at one of more locations accessible via URLs. If one of the listed locations defines a definitive reference implementation for the package definition then it MUST be listed as the first entry in the list.

The "ietf-yang-package" YANG module has the following structure:

```

module: ietf-yang-package
  +--ro yang-package
    +--ro name                yang:yang-identifier
    +--ro version              yang-sem-ver
    +--ro revision-date?      yanglib:revision-identifier
    +--ro location*            inet:uri
    +--ro description?         string
    +--ro reference?           string
    +--ro previous-version?    yang-sem-ver
    +--ro tag*                  tags:tag
    +--ro referentially-complete? boolean
    +--ro mandatory-feature*   string
    +--ro imported-packages* [name version]
      |   +--ro name            yang:yang-identifier
      |   +--ro version          yang-sem-ver
      |   +--ro deviated?       boolean
      |   +--ro location*        inet:uri
    +--ro module* [name]
      |   +--ro name            yang:yang-identifier
      |   +--ro revision?        revision-identifier
      |   +--ro version?         yang-sem-ver
      |   +--ro namespace        inet:uri
      |   +--ro location*        inet:uri
      |   +--ro submodule* [name]
      |     |   +--ro name            yang:yang-identifier
      |     |   +--ro revision        yanglib:revision-identifier
      |     |   +--ro location*      inet:uri
    +--ro import-only-module* [name revision]
      |   +--ro name            yang:yang-identifier
      |   +--ro revision          union
      |   +--ro version?         yang-sem-ver
      |   +--ro namespace        inet:uri
      |   +--ro location*        inet:uri
      |   +--ro submodule* [name]
      |     |   +--ro name            yang:yang-identifier
      |     |   +--ro revision        yanglib:revision-identifier
      |     |   +--ro location*      inet:uri
    +--ro replaces-revision*   yanglib:revision-identifier

```

## 8. YANG Packaging additions to YANG library

### 8.1. Package List

The main addition is a top level 'yang-library/package' list that lists all package of all versions known to the server. Each package itself is defined using imported packages and module-sets to define the specific set of modules implemented and imported by the package. The use of module-sets allows the module definitions to be shared with the existing YANG library schema definitions. The existing rule of RFC 7995bis related to combining modules-sets also applies here, i.e. The combined set of modules defined by the module-sets MUST NOT contain modules implemented at different revisions. I.e. the module-sets leaf-list is directly equivalent to the explicit module and import-only-module lists in the instance data YANG package definition.

The 'yang-library/package' list MAY include multiple versions of a particular package. E.g. if the server is capable of allowing clients to select which package versions should be used by the server.

### 8.2. Binding from schema to package

The second augmentation is to allow a server to optionally indicate that a schema definition directly relates to a package. Since YANG packages are available offline, it may be sufficient for a client to only check that a compatible version of the YANG package is being implemented by the server without fetching and comparing the full module list.

If a server indicates that its schema maps to a particular package then it MUST support all mandatory-features defined as part of that package, and it MUST NOT have any deviations to the modules defined by the package. A server MAY implement features not specified in the package's mandatory-features list.

If a server cannot faithfully implement a package then it can define a new package to accurately report what it does implement. The new package can include the original package as an imported package, and the new package can define additional modules containing deviations to the original package, allowing the new package to accurately describe the server behavior. There is no specific mechanism provided to indicate that a mandatory-feature is not supported on a server, but deviations MAY be used to disable functionality predicated by a mandatory-feature.

### 8.3. Tree diagram

The "ietf-yang-library-packages" YANG module has the following structure:

```

module: ietf-yang-library-packages
  augment /yanglib:yang-library:
    +--ro package* [name version]
      +--ro name          yang:yang-identifier
      +--ro version       yang-sem-ver
      +--ro revision-date? yanglib:revision-identifier
      +--ro location*     inet:uri
      +--ro description?  string
      +--ro reference?    string
      +--ro previous-version? yang-sem-ver
      +--ro tag*          tags:tag
      +--ro referentially-complete? boolean
      +--ro mandatory-feature* string
      +--ro imported-packages* [name version]
        +--ro name          yang:yang-identifier
        +--ro version       yang-sem-ver
        +--ro deviated?    boolean
      +--ro module-set*
        -> /yanglib:yang-library/module-set/name
  augment /yanglib:yang-library/yanglib:schema:
    +--ro package
      +--ro name?      -> /yanglib:yang-library/package/name
      +--ro version?   leafref
  augment /yanglib:yang-library/yanglib:module-set/
    yanglib:import-only-module:
    +--ro replaces-revision* yanglib:revision-identifier

```

### 9. YANG Packaging groupings

Groupings for YANG packaging related constructs are provided in a 'types' module for use by the instance-data and YANG library constructs described previously. They are also available to be used by other modules that have a need for packaging information.

The "ietf-yang-package-types" YANG module has the following structure:

```

module: ietf-yang-package-types

  grouping yang-pkg-identification-leafs

```



```

+----- name          yang:yang-identifier
+----- version      yang-sem-ver
grouping yang-pkg-common-leafs
+----- revision-date? yanglib:revision-identifier
+----- location*      inet:uri
+----- description?   string
+----- reference?     string
+----- previous-version? yang-sem-ver
+----- tag*           tags:tag
+----- referentially-complete? boolean
+----- mandatory-feature* string
+----- imported-packages* [name version]
|   +----- name          yang:yang-identifier
|   +----- version      yang-sem-ver
|   +----- deviated?    boolean
grouping yang-pkg-library-definition
+----- name          yang:yang-identifier
+----- version      yang-sem-ver
+----- revision-date? yanglib:revision-identifier
+----- location*      inet:uri
+----- description?   string
+----- reference?     string
+----- previous-version? yang-sem-ver
+----- tag*           tags:tag
+----- referentially-complete? boolean
+----- mandatory-feature* string
+----- imported-packages* [name version]
|   +----- name          yang:yang-identifier
|   +----- version      yang-sem-ver
|   +----- deviated?    boolean
+----- module-set*
|   -> /yanglib:yang-library/module-set/name
grouping yang-pkg-file-definition
+----- name          yang:yang-identifier
+----- version      yang-sem-ver
+----- revision-date? yanglib:revision-identifier
+----- location*      inet:uri
+----- description?   string
+----- reference?     string
+----- previous-version? yang-sem-ver
+----- tag*           tags:tag
+----- referentially-complete? boolean
+----- mandatory-feature* string
+----- imported-packages* [name version]
|   +----- name          yang:yang-identifier
|   +----- version      yang-sem-ver
|   +----- deviated?    boolean
|   +----- location*      inet:uri

```

```

+---- module* [name]
|   +---- name          yang:yang-identifier
|   +---- revision?     revision-identifier
|   +---- version?      yang-sem-ver
|   +---- namespace     inet:uri
|   +---- location*     inet:uri
|   +---- submodule* [name]
|       +---- name?      yang:yang-identifier
|       +---- revision   yanglib:revision-identifier
|       +---- location*  inet:uri
+---- import-only-module* [name revision]
|   +---- name?          yang:yang-identifier
|   +---- revision?      union
|   +---- version?       yang-sem-ver
|   +---- namespace     inet:uri
|   +---- location*     inet:uri
|   +---- submodule* [name]
|       +---- name?      yang:yang-identifier
|       +---- revision   yanglib:revision-identifier
|       +---- location*  inet:uri
+---- replaces-revision* yanglib:revision-identifier

```

## 10. YANG Modules

The YANG module definitions for the modules described in the previous sections.

```

<CODE BEGINS> file "ietf-yang-package-types@2018-11-26.yang"
module ietf-yang-package-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package-types";
  prefix "pkg-types";

  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-yang-library {
    prefix yanglib;
    reference "RFC 7895bis: YANG Library";
  }
  import ietf-module-tags {

```

```
    prefix tags;
    reference "XXX, (draft-ietf-netmod-module-tags-03): YANG Module Tags";
}

organization
  "IETF NETMOD (Network Modeling) Working Group";

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

  Author:     Rob Wilton
              <mailto:rwilton@cisco.com>";

description
  "This module provides type and grouping definitions for YANG
  packages.

  Copyright (c) 2018 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

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

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

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
revision 2018-11-26 {
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Schema Versioning.";
}

/*
 * Typedefs
 */

typedef yang-sem-ver {
  type string {
```

```
    pattern '\d+[.]\d+[.]\d+[mM]?';
  }
  description
    "Represents a YANG semantic version number.";
  reference
    "TODO - Should be defined by YANG versioning types module";
}

/*
 * Groupings
 */

grouping yang-pkg-identification-leafs {
  description
    "Parameters for identifying a specific version of a YANG
    package";

  leaf name {
    type yang:yang-identifier;
    mandatory true;
    description
      "The YANG package name.";
  }

  leaf version {
    type yang-sem-ver;
    mandatory true;
    description
      "YANG package version. Follows YANG semantic versions rules
      defined in XXX";
  }
}

grouping yang-pkg-common-leafs {
  description
    "Defines definitions common to all YANG package definitions.";

  leaf revision-date {
    type yanglib:revision-identifier;

    description
      "An optional revision identifier of when this package version
      was created. This does not need to be unique across all
      versions of a package.";
  }

  leaf-list location {
    type inet:uri;
  }
}
```

```
description
  "Contains a URL that represents where an instance data file
  for this YANG package can be found.

  This leaf will only be present if there is a URL
  available for retrieval of the schema for this entry.

  If multiple locations are provided, then the first location
  in the leaf-list MUST be the definitive location that
  uniquely identifies this package";
}

leaf description {
  type string;

  description "Provides a description of the package";
}

leaf reference {
  type string;

  description "Allows for a reference for the package";
}

leaf previous-version {
  type yang-sem-ver;
  description
    "The previous package version that this version has been
    derived from. This leaf allows a full version history graph
    to be constructed if required.";
}

leaf-list tag {
  type tags:tag;
  description
    "Tags associated with a YANG package. Module tags defined in
    XXX, ietf-netmod-module-tags can be used here but with the
    modification that the tag applies to the entire package
    rather than a specific module. See the IANA 'YANG Module Tag
    Prefix' registry for reserved prefixes and the IANA 'YANG
    Module IETF Tag' registry for IETF standard tags.";
}

leaf referentially-complete {
  type boolean;
  default true;
  description
    "Indicates whether the schema defined by this package is
```

```
    referentially complete. I.e. all module imports can be
    resolved to a module explicitly defined in this package or one
    of the imported packages.";
}

leaf-list mandatory-feature {
    type string;
    // TODO - Is there a better type for this?
    description
        "List all features from modules included in the package that
        MUST be supported by any server implementing the package.
        All other features defined in included packages are OPTIONAL
        to implement.

        Features are identified using <module-name>:<feature>";
}

list imported-packages {
    key "name version";
    description
        "An entry in this list represents a package that is imported
        as part of the package definition.

        If packages implement different revisions or versions of the
        same module, then an explicit entry in the module list MUST
        be provided to select the specific module version
        'implemented' by this package definition.

        For import-only modules, the replaces-revision leaf-list can
        be used to select the specific module versions imported by
        this package.";
    reference
        "XXX";

    uses yang-pkg-identification-leafs;

    leaf deviated {
        type boolean;
        default true;
        description
            "Set to true if any data nodes in this package are modified
            in a non backwards compatible way, either through the use
            of deviations, or because one of the modules has been
            replaced by an earlier module version.";
    }
}
}
```

```
grouping yang-pkg-file-definition {
  description
    "The set of parameters that describe a particular YANG package.";

  uses yang-pkg-identification-leafs;

  uses yang-pkg-common-leafs {
    augment "imported-packages" {
      description "Add the package location path";

      leaf-list location {
        type inet:uri;
        description
          "Contains a URL that represents where an instance data
           file for this YANG package can be found.

           This leaf will only be present if there is a URL
           available for retrieval of the schema for this entry.

           If multiple locations are provided, then the first
           location in the leaf-list MUST be the definitive location
           that uniquely identifies this package";
      }
    }
  }
}

list module {
  key "name";
  description
    "An entry in this list represents a module that must be
     implemented by a server implementing this package, as per
     RFC 7950 section 5.6.5, with a particular set of supported
     features and deviations.

     A entry in this list overrides any module version
     'implemented' by an imported package";
  reference
    "RFC 7950: The YANG 1.1 Data Modeling Language.";

  uses yanglib:module-identification-leafs;

  leaf version {
    type yang-sem-ver;
    description
      "The YANG module or submodule version.  If no version
       statement is present in the YANG module or submodule, this
       leaf is not instantiated.";
  }
}
```

```
leaf namespace {
  type inet:uri;
  mandatory true;
  description
    "The XML namespace identifier for this module.";
}
uses yanglib:location-leaf-list;

list submodule {
  key "name";
  description
    "Each entry represents one submodule within the
    parent module.";

  leaf name {
    type yang:yang-identifier;
    description
      "The YANG submodule name.";
  }
  leaf revision {
    type yanglib:revision-identifier;
    mandatory true;
    description
      "The YANG submodule revision date.  If the parent module
      include statement for this submodule includes a revision
      date then it MUST match this leaf's value.";
  }

  uses yanglib:location-leaf-list;
}

list import-only-module {
  key "name revision";
  description
    "An entry in this list indicates that the server imports
    reusable definitions from the specified revision of the
    module, but does not implement any protocol accessible
    objects from this revision.

    Multiple entries for the same module name MAY exist.  This
    can occur if multiple modules import the same module, but
    specify different revision-dates in the import statements.";

  leaf name {
    type yang:yang-identifier;
    description
      "The YANG module name.";
```



```
}
leaf revision {
  type union {
    type yanglib:revision-identifier;
    type string {
      length 0;
    }
  }
  description
    "The YANG module revision date. A zero-length string is
    used if no revision statement is present in the YANG
    module.";
}
leaf version {
  type yang-sem-ver;
  description
    "The YANG module or submodule version. If no version
    statement is present in the YANG module or submodule, this
    leaf is not instantiated.";
}
leaf namespace {
  type inet:uri;
  mandatory true;
  description
    "The XML namespace identifier for this module.";
}

uses yanglib:location-leaf-list;

list submodule {
  key "name";
  description
    "Each entry represents one submodule within the
    parent module.";

  leaf name {
    type yang:yang-identifier;
    description
      "The YANG submodule name.";
  }
  leaf revision {
    type yanglib:revision-identifier;
    mandatory true;
    description
      "The YANG submodule revision date. If the parent module
      include statement for this submodule includes a revision
      date then it MUST match this leaf's value.";
  }
}
```

```
        uses yanglib:location-leaf-list;
    }

    leaf-list replaces-revision {
        type yanglib:revision-identifier;
        description
            "Gives the revision of an import-only-module defined in
            an imported package that is replaced by this
            import-only-module revision.";
    }
}

grouping yang-pkg-library-definition {
    description
        "The set of parameters that describe a particular YANG package.";

    uses yang-pkg-identification-leafs;
    uses yang-pkg-common-leafs;

    leaf-list module-set {
        type leafref {
            path "/yanglib:yang-library/yanglib:module-set/yanglib:name";
        }
        description
            "Describes any modules in addition to, and replacing, and
            modules defined in the imported packages.

            If a non import-only module appears in multiple module sets,
            then the module revision and the associated features and
            deviations must be identical.";
    }
}
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-yang-package2018-11-26.yang"
module ietf-yang-package {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-package";
    prefix pkg;

    import ietf-yang-package-types {
        prefix pkg-types;
        reference "RFC XXX: YANG Schema Versioning.";
    }
}
```

## organization

"IETF NETMOD (Network Modeling) Working Group";

## contact

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

Author: Rob Wilton  
<<mailto:rwilton@cisco.com>>;

## description

"This module provides a definition of a YANG package, which is used as the schema for an YANG instance data document specifying a YANG package.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

// RFC Ed.: update the date below with the date of RFC publication  
// and remove this note.  
// RFC Ed.: replace XXXX with actual RFC number and remove this  
// note.

```
revision 2018-11-26 {  
  description  
    "Initial revision";  
  reference  
    "RFC XXXX: YANG Schema Versioning."  
}
```

```
/*  
 * Top-level container  
 */
```

```
container yang-package {  
  config false;  
  description  
    "Defines a YANG package.
```

Intended to be used to specify a YANG package as an instance data document.";

```
    uses pkg-types:yang-pkg-file-definition;
  }
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-yang-library-packages@2018-11-26.yang"
module ietf-yang-library-packages {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-yang-library-packages";
  prefix pkg;

  import ietf-yang-package-types {
    prefix pkg-types;
    reference "RFC XXX: YANG Packages.";
  }
  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 7895bis: YANG Library";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

    Author:   Rob Wilton
              <mailto:rwilton@cisco.com>";

  description
    "This module provides defined augmentations to YANG library to
    allow a server to report YANG package information.

    Copyright (c) 2018 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.
```

```
    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
```

```
(http://trustee.ietf.org/license-info).
```

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

```
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
revision 2018-11-26 {
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Schema Versioning.";
}

/*
 * Add in the list of packaged into YANG library.
 */
augment "/yanglib:yang-library" {
  description "Add YANG package definitions into YANG library";

  list package {
    config "false";
    key "name version";

    description "Defines the packages available on this server.";

    uses "pkg-types:yang-pkg-library-definition";
  }
}

/*
 * Allow schema to be related to a YANG package.
 */
augment "/yanglib:yang-library/yanglib:schema" {
  description
    "Allow datastore schema to be related to a YANG package";

  container package {
    leaf name {
      type leafref {
        path "/yanglib:yang-library/package/name";
      }
      description
        "The name of the package this schema relates to.";
    }
  }
}
```

```

    leaf version {
      type leafref {
        path '/yanglib:yang-library/'
          + 'package[name = current()/../name]/version';
      }

      description
        "The version of the package this schema relates to.";
    }

    description
      "Describes which package the schema directly relates to, if
        any.";
  }
}

/*
 * Allow import-only modules to list the versions that they are
 * replacing.
 */

augment
  "/yanglib:yang-library/yanglib:module-set/" +
  "yanglib:import-only-module" {

    description
      "Add replaces-revision to import-only-module definitions";

    leaf-list replaces-revision {
      type yanglib:revision-identifier;
      description
        "Gives the revision of an import-only-module defined in an
          imported package that is replaced by this import-only-module
          revision.

          Only used for YANG package definitions";
    }
  }
}
<CODE ENDS>

```

## 11. Security Considerations

The YANG modules specified in this document defines a schema for data that is accessed by network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure

transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

Similarly to YANG library [I-D.ietf-netconf-rfc7895bis], some of the readable data nodes in these YANG modules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes.

One additional key different to YANG library, is that the 'ietf-yang-package' YANG module defines a schema to allow YANG packages to be defined in YANG instance data documents, that are outside the security controls of the network management protocols. Hence, it is important to also consider controlling access to these package instance data documents to restrict access to sensitive information.

As per the YANG library security considerations, the module, revision and version information in YANG packages may help an attacker identify the server capabilities and server implementations with known bugs since the set of YANG modules supported by a server may reveal the kind of device and the manufacturer of the device. Server vulnerabilities may be specific to particular modules, module revisions, module features, or even module deviations. For example, if a particular operation on a particular data node is known to cause a server to crash or significantly degrade device performance, then the packaging information will help an attacker identify server implementations with such a defect, in order to launch a denial-of-service attack on the device.

## 12. IANA Considerations

It is expected that a central registry of standard YANG package definitions is required to support this packaging solution.

It is unclear whether an IANA registry is also required to manage specific package versions. It is highly desirable to have a specific canonical location, under IETF control, where the definitive YANG package versions can be obtained from.

TODO - Add IANA registrations for YANG modules defined in this draft.

### 13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at <https://github.com/rgwilton/YANG-Packages-Draft/issues/>

### 14. Acknowledgements

Feedback helping shape this document has kindly been provided by Andy Bierman, Ladislav Lhotka, and Jason Sterne.

### 15. References

#### 15.1. Normative References

- [I-D.ietf-netconf-rfc7895bis]  
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.
- [I-D.ietf-netmod-module-tags]  
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-07 (work in progress), March 2019.
- [I-D.ietf-netmod-yang-instance-file-format]  
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-02 (work in progress), February 2019.
- [I-D.verdt-netmod-yang-versioning-reqs]  
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.



- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [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>>.
- [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>>.
- [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>>.

## 15.2. Informative References

- [I-D.bierman-netmod-yang-package] Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.
- [I-D.ietf-netmod-artwork-folding] Watsen, K., Wu, Q., Farrel, A., and B. Claise, "Handling Long Lines in Artwork in Internet-Drafts and RFCs", draft-ietf-netmod-artwork-folding-00 (work in progress), November 2018.
- [openconfigsemver] "Semantic Versioning for Openconfig Models", <<http://www.openconfig.net/docs/semver/>>.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.

## Appendix A. Tree output for ietf-yang-library with package augmentations

Complete tree output for ietf-yang-library with package augmentations.

```

module: ietf-yang-library
  +--ro yang-library
    |
    | +--ro module-set* [name]
    | |
    | | +--ro name string
    | | +--ro module* [name]
    | | |
    | | | +--ro name yang:yang-identifier
    | | | +--ro revision? revision-identifier
    | | | +--ro namespace inet:uri
    | | | +--ro location* inet:uri
    | | | +--ro submodule* [name]
    | | | |
    | | | | +--ro name yang:yang-identifier
    | | | | +--ro revision? revision-identifier
    | | | | +--ro location* inet:uri
    | | | +--ro feature* yang:yang-identifier
    | | | +--ro deviation* -> ../../module/name
    | | +--ro import-only-module* [name revision]
    | | |
    | | | +--ro name yang:yang-identifier
    | | | +--ro revision union
    | | | +--ro namespace inet:uri
    | | | +--ro location* inet:uri
    | | | +--ro submodule* [name]
    | | | |
    | | | | +--ro name yang:yang-identifier
    | | | | +--ro revision? revision-identifier
    | | | | +--ro location* inet:uri
    | | | +--ro pkg:replaces-revision*
    | | | | yanglib:revision-identifier
    | +--ro schema* [name]
    | |
    | | +--ro name string
    | | +--ro module-set* -> ../../module-set/name
    | | +--ro pkg:package
    | | | +--ro pkg:name?
    | | | | -> /yanglib:yang-library/package/name
    | | | +--ro pkg:version? leafref
    | +--ro datastore* [name]
    | |
    | | +--ro name ds:datastore-ref
    | | +--ro schema -> ../../schema/name
    | +--ro content-id string
    | +--ro pkg:package* [name version]
    | |
    | | +--ro pkg:name yang:yang-identifier
    | | +--ro pkg:version yang-sem-ver
    | | +--ro pkg:revision-date?
    | | | yanglib:revision-identifier
    | | +--ro pkg:location* inet:uri
    | | +--ro pkg:description? string

```

```

|      +--ro pkg:reference?                string
|      +--ro pkg:previous-version?        yang-sem-ver
|      +--ro pkg:tag*                     tags:tag
|      +--ro pkg:referentially-complete?  boolean
|      +--ro pkg:mandatory-feature*       string
|      +--ro pkg:imported-packages* [name version]
|      |      +--ro pkg:name                yang:yang-identifier
|      |      +--ro pkg:version            yang-sem-ver
|      |      +--ro pkg:deviated?          boolean
|      +--ro pkg:module-set*
|          -> /yanglib:yang-library/module-set/name
x--ro modules-state
  x--ro module-set-id    string
  x--ro module* [name revision]
    x--ro name                yang:yang-identifier
    x--ro revision            union
    +--ro schema?             inet:uri
    x--ro namespace          inet:uri
    x--ro feature*            yang:yang-identifier
    x--ro deviation* [name revision]
      |      x--ro name                yang:yang-identifier
      |      x--ro revision            union
    x--ro conformance-type    enumeration
    x--ro submodule* [name revision]
      x--ro name                yang:yang-identifier
      x--ro revision            union
      +--ro schema?             inet:uri

notifications:
  +---n yang-library-update
  |   +--ro content-id    -> /yang-library/content-id
  x---n yang-library-change
    x--ro module-set-id    -> /modules-state/module-set-id

```

## Appendix B. Examples

This section provides various examples of YANG packages, and as such this text is non-normative. The purpose of the examples is to only illustrate the file format of YANG packages, and how package dependencies work. It does not imply that such packages will be defined by IETF, or which modules would be included in those packages even if they were defined.

## B.1. Example IETF Network Device YANG package

This section provides an instance data document example of an IETF Network Device YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, to implement a basic network device without any dynamic routing or layer 2 services. E.g., it includes functionality such as system information, interface and basic IP configuration.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number.

```
<CODE BEGINS> file "example-ietf-network-device-pkg.json"
===== NOTE: '\\\'' line wrapping per BCP XX (RFC XXXX) =====

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-network-device-pkg",
    "target-ptr": "TBD",
    "timestamp": "2018-12-13T17:00:00Z",
    "description": "Example IETF network device YANG package definiti\
\ion",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-ietf-network-device",
        "version": "1.1.2",
        "namespace": "urn:ietf:params:xml:ns:yang-pkg:ietf-network-d\
\evice",
        "location": "file://example.org/yang/packages/ietf-network-d\
\evice@v1.1.2.json",
        "description": "This package defines a small sample set of Y\
\ANG modules that could represent the basic set of modules that a st\
\andard network device might be expected to support.",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-11-26",
        "module": [
          {
            "name": "iana-crypt-hash",
            "revision": "2014-08-06",
            "namespace": "urn:ietf:params:xml:ns:yang:iana-crypt-has\
\h",
            "location": "https://raw.githubusercontent.com/YangModel\
\s/yang/master/standard/ietf/RFC/iana-crypt-hash%402014-08-06.yang"
```

```

    },
    {
      "name": "ietf-system",
      "revision": "2014-08-06",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-system",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-system%402014-08-06.yang"
    },
    {
      "name": "ietf-interfaces",
      "revision": "2018-02-20",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-interface\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-interfaces%402018-02-20.yang"
    },
    {
      "name": "ietf-netconf-acm",
      "revision": "2018-02-14",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-netconf-a\
\cm",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-netconf-acm%402018-02-14.yang"
    },
    {
      "name": "ietf-key-chain",
      "revision": "2017-06-15",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-key-chain\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-key-chain%402017-06-15.yang"
    },
    {
      "name": "ietf-ip",
      "revision": "2018-02-22",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-ip",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-ip%402018-02-22.yang"
    }
  ],
  "import-only-module": [
    {
      "name": "ietf-yang-types",
      "revision": "2013-07-15",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-yang-type\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/ietf/master/standard/ietf/RFC/ietf-yang-types%402013-07-15.yang"
    }
  ]
}

```

```

    },
    {
      "name": "ietf-inet-types",
      "revision": "2013-07-15",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-inet-type\
\s",
      "location": "https://raw.githubusercontent.com/YangModel\
\s/yang/master/standard/ietf/RFC/ietf-inet-types%402013-07-15.yang"
    }
  ]
}
}
}
}
<CODE ENDS>

```

## B.2. Example IETF Basic Routing YANG package

This section provides an instance data document example of a basic IETF Routing YANG package formatted in JSON.

This example package is intended to represent the standard set of YANG modules, with import dependencies, that builds upon the example-ietf-network-device YANG package to add support for basic dynamic routing and ACLs.

As for all YANG packages, all import dependencies are fully resolved. Because this example uses YANG modules that have been standardized before YANG semantic versioning, they modules are referenced by revision date rather than version number. Locations have been excluded where they are not currently known, e.g., for YANG modules defined in IETF drafts. In a normal YANG package, locations would be expected to be provided for all YANG modules.

```

<CODE BEGINS> file "example-ietf-routing-pkg.json"
===== NOTE: '\n' line wrapping per BCP XX (RFC XXXX) =====

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-ietf-routing-pkg",
    "target-ptr": "TBD",
    "timestamp": "2018-12-13T17:00:00Z",
    "description": "Example IETF routing YANG package definition",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-ietf-routing",

```

```

    "version": "1.3.1",
    "namespace": "urn:ietf:params:xml:ns:yang-pkg:ietf-routing",
    "location": "file://example.org/yang/packages/ietf-routing@v\
1.3.1.json",
    "description": "This package defines a small sample set of I\
ETF routing YANG modules that could represent the set of IETF routi\
ng functionality that a basic IP network device might be expected t\
o support.",
    "reference": "XXX, draft-rwilton-netmod-yang-packages",
    "revision-date": "2018-11-26",
    "imported-packages": [
      {
        "name": "ietf-network-device",
        "version": "1.1.2",
        "location": [
          "http://example.org/yang/packages/ietf-network-device@\
v1.1.2.json"
        ]
      }
    ],
    "module": [
      {
        "name": "ietf-routing",
        "revision": "2018-03-13",
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing",
        "location": [
          "https://raw.githubusercontent.com/YangModels/yang/mas\
ter/standard/ietf/RFC/ietf-routing@2018-03-13.yang"
        ]
      },
      {
        "name": "ietf-ipv4-unicast-routing",
        "revision": "2018-03-13",
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-ipv4-unca\
st-routing",
        "location": [
          "https://raw.githubusercontent.com/YangModels/yang/mas\
ter/standard/ietf/RFC/ietf-ipv4-unicast-routing@2018-03-13.yang"
        ]
      },
      {
        "name": "ietf-ipv6-unicast-routing",
        "revision": "2018-03-13",
        "namespace": "urn:ietf:params:xml:ns:yang:ietf-ipv6-unca\
st-routing",
        "location": [
          "https://raw.githubusercontent.com/YangModels/yang/mas\
ter/standard/ietf/RFC/ietf-ipv6-unicast-routing@2018-03-13.yang"
        ]
      }
    ]
  }
}

```

```

    ]
  },
  {
    "name": "ietf-isis",
    "revision": "2018-12-11",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-isis"
  },
  {
    "name": "ietf-interfaces-common",
    "revision": "2018-07-02",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-interface\
\s-common"
  },
  {
    "name": "ietf-if-l3-vlan",
    "revision": "2017-10-30",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-if-l3-vla\
\n"
  },
  {
    "name": "ietf-routing-policy",
    "revision": "2018-10-19",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing-p\
\olicy"
  },
  {
    "name": "ietf-bgp",
    "revision": "2018-05-09",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-bgp"
  },
  {
    "name": "ietf-access-control-list",
    "revision": "2018-11-06",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-access-co\
\ntrol-list"
  }
],
"import-only-module": [
  {
    "name": "ietf-routing-types",
    "revision": "2017-12-04",
    "namespace": "urn:ietf:params:xml:ns:yang:ietf-routing-t\
\ypes",
    "location": [
      "https://raw.githubusercontent.com/YangModels/yang/master/standard/ietf/RFC/ietf-routing-types@2017-12-04.yang"
    ]
  }
],

```



```

        {
            "name": "iana-routing-types",
            "revision": "2017-12-04",
            "namespace": "urn:ietf:params:xml:ns:yang:iana-routing-t\
\ypes",
            "location": [
                "https://raw.githubusercontent.com/YangModels/yang/mas\
\ter/standard/ietf/RFC/iana-routing-types@2017-12-04.yang"
            ]
        },
        {
            "name": "ietf-bgp-types",
            "revision": "2018-05-09",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-bgp-types"
        },
        {
            "name": "ietf-packet-fields",
            "revision": "2018-11-06",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-packet-fi\
\elds"
        },
        {
            "name": "ietf-ethertypes",
            "revision": "2018-11-06",
            "namespace": "urn:ietf:params:xml:ns:yang:ietf-ethertype\
\s"
        }
    ]
}
}
}
}
<CODE ENDS>

```

### B.3. Package import conflict resolution example

This section provides an example of how a package can resolve conflicting module versions from imported packages.

In this example, YANG package 'example-3-pkg' imports both 'example-import-1' and 'example-import-2' packages. However, the two imported packages implement different versions of 'example-module-A' so the 'example-3-pkg' package selects version '1.2.3' to resolve the conflict. Similarly, for import-only modules, the 'example-3-pkg' package does not require both versions of example-types-module-C to be imported, so it indicates that it only imports revision '2018-11-26' and not '2018-01-01'.

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-1-pkg",
    "description": "First imported example package",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-import-1",
        "version": "1.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-01-01",
        "module": [
          {
            "name": "example-module-A",
            "version": "1.0.0"
          },
          {
            "name": "example-module-B",
            "version": "1.0.0"
          }
        ],
        "import-only-module": [
          {
            "name": "example-types-module-C",
            "revision": "2018-01-01"
          },
          {
            "name": "example-types-module-D",
            "revision": "2018-01-01"
          }
        ]
      }
    }
  }
}

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-import-2-pkg",
    "description": "Second imported example package",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-import-2",
        "version": "2.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-11-26",
        "module": [
          {
            "name": "example-module-A",
```

```
        "version": "1.2.3"
      },
      {
        "name": "example-module-E",
        "version": "1.1.0"
      }
    ],
    "import-only-module": [
      {
        "name": "example-types-module-C",
        "revision": "2018-11-26"
      },
      {
        "name": "example-types-module-D",
        "revision": "2018-11-26"
      }
    ]
  }
}

{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "example-3-pkg",
    "description": "Importing example package",
    "content-data": {
      "ietf-yang-package:yang-package": {
        "name": "example-3",
        "version": "1.0.0",
        "reference": "XXX, draft-rwilton-netmod-yang-packages",
        "revision-date": "2018-11-26",
        "imported-packages": [
          {
            "name": "example-import-1",
            "version": "1.0.0"
          },
          {
            "name": "example-import-2",
            "version": "2.0.0"
          }
        ],
        "module": [
          {
            "name": "example-module-A",
            "version": "1.2.3"
          }
        ]
      }
    }
  }
}
```

```
    "import-only-module": [  
      {  
        "name": "example-types-module-C",  
        "revision": "2018-11-26",  
        "replaces-revision": [ "2018-01-01 " ]  
      }  
    ]  
  }  
}  
}
```

## Author's Address

Robert Wilton  
Cisco Systems, Inc.

Email: [rwilton@cisco.com](mailto:rwilton@cisco.com)

Network Working Group  
Internet-Draft  
Updates: 7950 (if approved)  
Intended status: Standards Track  
Expires: September 12, 2019

B. Claise  
J. Clarke  
R. Rahman  
R. Wilton, Ed.  
Cisco Systems, Inc.  
B. Lengyel  
Ericsson  
J. Sterne  
Nokia  
K. D'Souza  
AT&T  
March 11, 2019

YANG Semantic Versioning for Modules  
draft-verdt-netmod-yang-semver-00

Abstract

This document specifies a new YANG module update procedure using semantic version numbers, to allow for limited non-backwards-compatible changes, as an alternative proposal to module update rules in the YANG 1.1 specifications. This document updates RFC 7950, RFC 8407 and RFC 8525.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1.	Introduction . . . . .	3
1.1.	Updates to YANG RFCs . . . . .	4
1.1.1.	Updates to RFC7950 . . . . .	4
1.1.2.	Updates to RFC8525 . . . . .	4
1.1.3.	Updates to RFC8407 . . . . .	5
1.2.	Complementary solutions for the other requirements . . . . .	5
2.	YANG Semantic Versioning . . . . .	6
2.1.	Classification of changes between module revisions . . . . .	6
2.2.	YANG Semantic Versioning Scheme for Modules . . . . .	7
2.2.1.	Examples for YANG semantic version numbers . . . . .	8
2.3.	YANG Semantic Version Update Rules . . . . .	10
2.4.	YANG Module Semver Extension . . . . .	11
3.	Import by Semantic Version . . . . .	13
3.1.	Module import examples . . . . .	15
4.	Classifying changes in YANG modules . . . . .	16
4.1.	Editorial changes . . . . .	16
4.2.	Backwards-compatible changes . . . . .	16
4.3.	Non-backwards-compatible changes . . . . .	17
5.	Updates to ietf-yang-library . . . . .	17
5.1.	Advertising module version number . . . . .	17
5.2.	Resolving ambiguous module imports . . . . .	17
5.3.	Reporting how deprecated and obsolete nodes are handled . . . . .	18
6.	YANG status description extension . . . . .	19
7.	Semantic versioning of YANG instance data . . . . .	19
8.	Guidelines . . . . .	20
8.1.	Guidelines to YANG model authors . . . . .	20
8.1.1.	Use of YANG semantic versioning . . . . .	20
8.1.2.	Making non-backwards-compatible changes to a YANG module . . . . .	21
8.1.2.1.	Removing a data node . . . . .	22
8.1.2.2.	Changing the type of a leaf node . . . . .	23
8.1.2.3.	Reducing the range of a leaf node . . . . .	23
8.1.2.4.	Changing the key of a list . . . . .	23
8.1.2.5.	Renaming a node . . . . .	23
8.1.2.6.	Changing a default value . . . . .	23
8.2.	Guidelines to YANG model clients . . . . .	24

9. Semantic Version Extension YANG Modules . . . . .	24
10. Contributors . . . . .	30
11. Security Considerations . . . . .	31
12. IANA Considerations . . . . .	31
12.1. YANG Module Registrations . . . . .	31
13. References . . . . .	32
13.1. Normative References . . . . .	32
13.2. Informative References . . . . .	32
Appendix A. Appendix . . . . .	34
A.1. Open Issues . . . . .	34
A.2. Derived Semantic Version . . . . .	35
A.2.1. The Derived Semantic Version . . . . .	35
A.2.2. Implementation Experience . . . . .	35
Authors' Addresses . . . . .	36

## 1. Introduction

This document defines a solution to the YANG module lifecycle problems described in [I-D.verdt-netmod-yang-versioning-reqs], covering all of the specified requirements except for requirements: 2.2, 3.1, and 3.2.

Specifically, this document recognises a need to sometimes allow YANG modules to evolve with non-backwards-compatible changes, which might end up breaking clients. The solution makes use of semantic version numbers to help manage the lifecycle of YANG modules.

The solution is comprised of the following seven parts:

- A definition for the YANG semantic versioning scheme for modules, and an explanation of how the semver extension can be used to annotate modules with their semantic version number.

- A YANG extension to allow YANG module imports to be restricted to modules with particular semantic versions, allowing inter-module version dependencies to be captured within YANG module definitions.

- Updates to the YANG 1.1 module update rules to accommodate the semantic versioning scheme.

- Updates and augmentations to ietf-yang-library to include the YANG semantic version number in the module descriptions, to report how 'deprecated' and 'obsolete' nodes are handled by a server, and to clarify how module imports are resolved when multiple versions could otherwise be chosen.

A YANG extension to add a 'description' statement to the YANG 'status' statement to allow additional documentation as to why a node is being deprecated, and what alternatives may be available.

A description of how YANG semantic versioning applies to YANG instance data.

Guidelines to YANG module authors on how the YANG semantic versioning rules should be used, along with examples.

Open issues are listed at Appendix A.1, and tracked at <https://github.com/netmod-wg/yang-ver-dt/issues>.

## 1.1. Updates to YANG RFCs

### 1.1.1. Updates to RFC7950

This document proposes updates to [RFC7950] to address some of the requirements. It should be noted that there is also active WG discussion on the next steps towards an updated version of YANG, and potentially some of the functionality described here could be folded into an updated revision of [RFC7950], although that might adversely impact when (parts of) a standards based YANG module versioning solution is available.

The sections listed below provide updates to [RFC7950]. The design team does not believe any of the changes require a new version of the YANG language. It is believed that the extensions as they are defined can coexist with existing YANG 1.1 clients.

- o Section 4 describes modification to the [RFC7950] Section 11 module update text to advise the use of semantic versioning as described in this document.
- o Section 3 describes an extension to do import by semantic version.
- o Section 6 defines an extension that adds a description child element to the YANG "status" statement.

### 1.1.2. Updates to RFC8525

This document updates [RFC8525]. Section 5 defines how a reader of a YANG library datastore schema chooses which version of an import-only module is used to resolve a module import when the definition is otherwise ambiguous.



### 1.1.3. Updates to RFC8407

Section 8 updates [RFC8407] to provide guidelines on how the YANG module semantic versioning can be used to manage the lifecycle of YANG modules when using strict RFC 7950 chapter 11 backwards compatibility rules are not pragmatic.

### 1.2. Complementary solutions for the other requirements

This section is to aid the WG understand how the full set of YANG versioning requirements are intended to be holistically addressed and is intended to be removed if this draft is adopted by the WG.

As stated previously, this draft does not address requirements 2.2, 3.1 and 3.2 of the requirements specified in [I-D.verdt-netmod-yang-versioning-reqs]. Instead, additional work is needed to address those requirements, which the design team believes would be best addressed in separate drafts. It is hoped that the WG agrees that viable solutions to the other requirements exist that complement the solution proposed in this draft, and thus this work can usefully progress in parallel. In particular, there is value to the industry to achieve standardization of a partial solution that addresses the majority, but not all, of the stated requirements, on the agreement that a full solution will follow.

The two additional drafts are:

A tooling based solution is proposed for requirement 2.2, that allows two YANG schema versions to be algorithmically compared, with the algorithm reporting the list of differences between the two YANG schema and whether each change is regarded as being editorial, backwards-compatible, or non-backwards-compatible. Annotations to the YANG modules, via the use of extension statements, may help improve the accuracy of the comparison algorithm, particularly for statements that are very hard for an algorithm to correctly classify the scope of any differences (e.g., a change in the semantic behaviour of a data node defined via modifications to the associated YANG description statement). Given that requirement 2.2 is a soft requirement (SHOULD rather than MUST), and practical experience with the tooling is required, it is proposed that this work is deferred at this time.

A proposed solution for requirements 3.1 and 3.2 is via the use of YANG packages [I-D.rwilton-netmod-yang-packages] and a protocol based version selection scheme that can be used by clients to choose a particular YANG datastore schema from the set of datastore schema that are supported by the server.

## 2. YANG Semantic Versioning

The chapter defines YANG Semantic Versioning, explains how it is used with YANG modules, and the rules associated with changing a module's semantic version number when the module definitions are updated.

The YANG semantic versioning scheme applies only to YANG modules. YANG submodules are not independently versioned by the YANG semantic versioning scheme. Instead, if a versioned module includes one or more submodules then those submodules are implicitly versioned as part of the module's 'semver:version' statements, and all the module's 'include' statements MUST specify the revision-date for each of the included submodules.

### 2.1. Classification of changes between module revisions

The principle aim of YANG semantic versioning is to allow a user of a YANG module to understand the overall significance of any changes between two module revisions solely based on the semantic version number.

The semantic version change between any two arbitrary revisions of a YANG module can be classified into one of four categories: 'unchanged', 'editorial', 'backwards-compatible' or 'non-backwards-compatible'. A summary of the classification is given below, with the specific rules as they apply to YANG statements provided in Section 4.

The semantic version change between two module revisions is defined as 'unchanged' if, after excluding 'revision' and 'semver:version' statements and their substatements, the only remaining changes are insignificant white space changes.

An 'editorial' module semantic version change is where there are changes in the module's statements, between the two module revisions, but those changes do not affect the syntax or semantic meaning of the module in any way. An example of an editorial change would be a fix to a spelling mistake in a description statement.

A 'backwards-compatible' module semantic version change is where some syntax or semantic changes exists between the two module revisions, but all changes follow the rules specified in Section 4.2.

A 'non-backwards-compatible' module semantic version change is where some syntax or semantic changes exists between the two

module revisions, and those changes do not follow the rules for a 'backwards-compatible' version change.

## 2.2. YANG Semantic Versioning Scheme for Modules

This document defines the YANG semantic versioning scheme that is used for YANG modules. The versioning scheme has the following properties:

The YANG semantic versioning scheme is extended from version 2.0.0 of the semantic versioning scheme defined at [semver.org](http://semver.org) [semver] to cover the additional requirements for the management of YANG module lifecycles that cannot be addressed using the [semver.org](http://semver.org) 2.0.0 versioning scheme alone.

Unlike the [semver.org](http://semver.org) 2.0.0 versioning scheme, the YANG semantic versioning scheme supports limited updates to older versions of YANG modules, to allow for bug fixes and enhancements to module versions that are not the latest.

Module definitions that follow the [semver.org](http://semver.org) 2.0.0 versioning scheme are fully compatible with implementations that understand the YANG semantic versioning scheme.

If module updates are always restricted to the latest version of the module only, then the version numbers used by the YANG semantic versioning scheme are exactly the same as those defined by the [semver.org](http://semver.org) 2.0.0 versioning scheme.

Every YANG module versioned using the YANG semantic versioning scheme specifies the module's semantic version number by including the 'semver:module-version' statement according to the following rules:

The module **MUST** include at least one revision statement.

The most recent module revision statement **MUST** include a 'semver:module-version' sub-statement, that defines the module's YANG semantic version.

The preceding module revision statement **SHOULD** also include a 'semver:module-version' sub-statement, to allow the module's semantic version history to be derived.

All other revision statements **MAY** include a 'semver:module-version' sub-statement if they have an associated YANG semantic version.

"The YANG semver version number is expressed as a string of the form: 'X.Y.Zv'; where X, Y, and Z each represent non-negative integers smaller than 32768, and v represents an optional single character suffix: 'm' or 'M'.

- o 'X' is the MAJOR version. Changes in the major version number indicate changes that are non-backwards-compatible to versions with a lower major version number.
- o 'Y' is the MINOR version. Changes in the minor version number indicate changes that are backwards-compatible to versions with the same major version number, but a lower minor version number and no patch 'm' or 'M' modifier.
- o 'Zv' is the PATCH version and modifier. Changes in the patch version number can indicate editorial, backwards-compatible, or non-backwards-compatible changes relative to versions with the same major and minor version numbers, but lower patch version number, depending on what form modifier 'v' takes:
  - \* If the modifier letter is absent, the change represents an editorial change
  - \* 'm' - the change represents a backwards-compatible change
  - \* 'M' - the change represents a non-backwards-compatible change

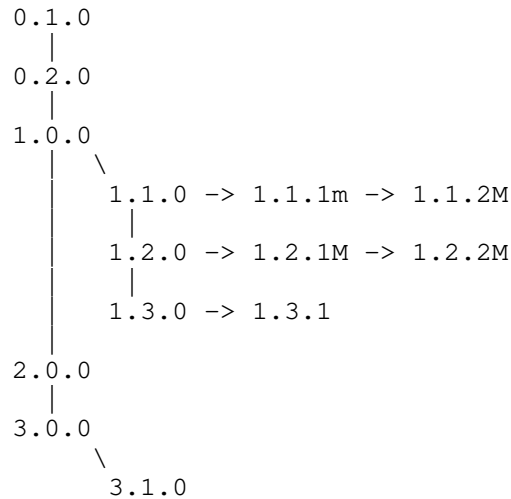
The YANG module name and YANG semantic version number uniquely identifies a revision of a module, with an associated revision date. There MUST NOT be multiple instances of a YANG module definition with the same module name and YANG semantic version number but different content or revision date.

There MUST NOT be multiple versions of a YANG module that have the same MAJOR, MINOR and PATCH version numbers, but different patch modifier letter. E.g., module version "1.2.3M" MUST NOT be defined if module version "1.2.3" has already been defined.

#### 2.2.1. Examples for YANG semantic version numbers

The following diagram and explanation illustrates how YANG semantic version numbers work.

Example YANG semantic version numbers for an example module:



The tree diagram above illustrates how an example modules version history might evolve. For example, the tree might represent the following changes, listed in chronological order from oldest revision to newest:

- 0.1.0 – first beta module version
- 0.2.0 – second beta module version (with NBC changes)
- 1.0.0 – first release (may have NBC changes from 0.2.0)
- 1.1.0 – added new functionality, leaf "foo" (BC)
- 1.2.0 – added new functionality, leaf "baz" (BC)
- 1.3.0 – improve existing functionality, added leaf "foo-64" (BC)
- 1.3.1 – improve description wording for "foo-64" (Editorial)
- 1.1.1m – backport "foo-64" leaf to 1.1.x to avoid implementing "baz" from 1.2.0 (BC)
- 2.0.0 – change existing model for performance reasons, e.g. re-key list (NBC)
- 1.1.2M – NBC point bug fix, not required in 2.0.0 due to model changes (NBC)

3.0.0 - NBC bugfix, rename "baz" to "bar"; also add new BC leaf "wibble"; (NBC)

1.2.1M - backport NBC fix, changing "baz" to "bar"

1.2.2M - backport "wibble". This is a BC change but "M" modifier is sticky.

3.1.0 - introduce new leaf "wobble" (BC)

The partial ordering relationships based on the semantic versioning numbers can be defined as follows:

1.0.0 < 1.1.0 < 1.2.0 < 1.3.0 < 2.0.0 < 3.0.0 < 3.1.0

1.0.0 < 1.1.0 < 1.1.1m < 1.1.2M

1.0.0 < 1.1.0 < 1.2.0 < 1.2.1M < 1.2.2M

There is no ordering relationship between 1.1.1M and either 1.2.0 or 1.2.1M, except that they share the common ancestor of 1.1.0.

Looking at the version number alone, the module definition in 2.0.0 does not necessarily contain the contents of 1.3.0. However, the module revision history in 2.0.0 would likely indicate that it was edited from module version 1.3.0.

## 2.3. YANG Semantic Version Update Rules

When a new revision of a module is produced, then the following rules define how the YANG semantic version number for the new module revision is calculated, based on the changes between the two module revisions, and the YANG semantic version number of the base module revision that the changes are derived from. A two step process is used:

The first step is to classify the module change as 'editorial', 'backwards-compatible', or 'non-backwards-compatible version' using the rules defined in Section 2.1 and Section 4.

The second step is to calculate the value of the 'semver:version' field for the new module revision, based on the value of the 'semver:version' field in the base module, any how the module changes have been classified.

The following rules define how the value for the 'semver:version' argument in the new module revision is calculated:

1. If a module is being updated in a non-backwards-compatible way, then the module version "X.Y.Z[m|M]" MUST be updated to "X+1.0.0" unless that module version has already been defined with different content, in which case the module version "X.Y.Z+1M" MUST be used instead.
2. If a module is being updated in a backwards-compatible way, then the next version number depends on the format of the current version number:
  - i "X.Y.Z" - the module version MUST be updated to "X.Y+1.0", unless that module version has already been defined with different content, when the module version MUST be updated to "X.Y.Z+1m instead".
  - ii "X.Y.Zm" - the module version MUST be updated to "X.Y.Z+1m".
  - iii "X.Y.ZM" - the module version MUST be updated to "X.Y.Z+1M".
3. If a module is being updated in an editorial way, then the next version number depends on the format of the current version number:
  - i "X.Y.Z" - the module version MUST be updated to "X.Y.Z+1"
  - ii "X.Y.Zm" - the module version MUST be updated to "X.Y.Z+1m".
  - iii "X.Y.ZM" - the module version MUST be updated to "X.Y.Z+1M".
4. YANG module semantic version numbers beginning with 0, i.e "0.X.Y" are regarded as beta definitions and need not follow the rules above. Either the MINOR or PATCH version numbers may be updated, regardless of whether the changes are non-backwards-compatible, backwards-compatible, or editorial.

#### 2.4. YANG Module Semver Extension

This document defines a YANG extension to add the YANG module semantic version to a Module. The complete definition of this YANG module is in Section 9.

```
extension module-version {  
    argument semver;  
}
```

The extension would typically be used this way:

```
module yang-module-name {  
  
    namespace "name-space";  
    prefix "prefix-name";  
  
    import ietf-semver { prefix "semver"; }  
  
    description  
        "to be completed";  
  
    revision 2018-02-28 {  
        description "Added leaf 'wobble'";  
        semver:module-version "3.1.0";  
    }  
  
    revision 2017-12-31 {  
        description "Rename 'baz' to 'bar', added leaf 'wibble'";  
        semver:module-version "3.0.0";  
    }  
  
    revision 2017-10-30 {  
        description "Change the module structure";  
        semver:module-version "2.0.0";  
    }  
  
    revision 2017-08-30 {  
        description "Clarified description of 'foo-64' leaf";  
        semver:module-version "1.3.1";  
    }  
  
    revision 2017-07-30 {  
        description "Added leaf foo-64";  
        semver:module-version "1.3.0";  
    }  
  
    revision 2017-04-20 {  
        description "Add new functionality, leaf 'baz'";  
        semver:module-version "1.2.0";  
    }  
  
    revision 2017-04-03 {  
        description "Add new functionality, leaf 'foo'";  
        semver:module-version "1.1.0";  
    }  
  
    revision 2017-04-03 {
```



```
    description "First release version.";
    semver:module-version "1.0.0";
  }

  revision 2017-01-30 {
    description "NBC changes to initial revision";
    semver:module-version "0.2.0";
  }

  revision 2017-01-26 {
    description "Initial module version";
    semver:module-version "0.1.0";
  }

  //YANG module definition starts here
```

See also "Semantic Versioning and Structure for IETF Specifications" [I-D.claise-semver] for a mechanism to combine the semantic versioning, the GitHub tools, and a potential change to the IETF process.

### 3. Import by Semantic Version

RFC 7950 allows YANG module 'import' statements to optionally require the imported module to have a particular revision date. In practice, importing a module with an exact revision date is overly burdensome because it requires the importing module to be updated whenever any change to the imported module occurs. The alternative choice of using an import statement without a revision date is also not ideal because the importing module may not work with all possible revisions of the imported module.

With semantic versioning, it is desirable for a importing module to specify the set of module versions of the imported module that are anticipated to be compatible.

This document specifies a YANG extension for selecting which versions of a module may be imported. It is designed around the assumption that most changes to a YANG module do not break importing modules, even if the changes themselves are not backwards compatible. E.g., fixing an incorrect pattern statement or description for a leaf would not break an import, changing the name of a leaf could break an import but frequently would not, but removing a container would break imports if it is augmented by another module.

The ietf-semver module defines the 'version' extension, a substatement to the YANG 'import' statement.

An 'import' statement MAY contain 'version' statements or a 'revision-date' statement, but not both.

The 'version' statement MAY be specified multiple times, requiring that the imported module version conforms to at least one of the 'version' statements.

The argument to the 'version' statement takes one of three valid forms:

1. "A.B.C" - import the exact module version that matches "A.B.C".
2. "A.B.C+" - import any module version that matches, or is greater than, "A.B.C".
3. "A.B.C-X.Y.Z" - import any module version that matches, or is greater than, "A.B.C"; and also matches, or is less than, "X.Y.Z". The word "MAX" can be used for 'Y' or 'Z' to represent the numerical value 32,767.

The rules for comparing module version numbers are as follows:

1. Version "R.S.T" matches version "A.B.C", only if
$$R = A, S = B, \text{ and } T = C$$
2. Version "R.S.T" is greater than version "A.B.C", only if
$$R = A, S = B, \text{ and } T > C; \text{ or}$$
$$R = A \text{ and } S > B; \text{ or}$$
$$R > A$$
3. Version "R.S.T" is less than version "X.Y.Z", only if
$$R = X, S = Y, \text{ and } T < Z; \text{ or}$$
$$R = X \text{ and } S < Y; \text{ or}$$
$$R < X$$

The patch modifier letter is not included as part of the 'semver:version' argument, and is entirely ignored for import statement module version number comparisons.

### 3.1. Module import examples

Consider an example module "example-module" that is hypothetically available in the following versions: 0.1.0, 0.2.0, 1.0.0, 1.1.0, 1.1.1m, 1.1.2M, 1.2.0, 1.2.1M, 1.2.2M, 1.3.0, 1.3.1, 2.0.0, 3.0.0, and 3.1.0. E.g. matching the versions illustrated in Section 2.2.1.

The first example selects the specific version 1.1.2M. A specific version import might be used if 1.1.2M contained changes that are incompatible with other versions.

```
import example-module {  
    semver:version 1.1.2;  
}
```

The next example selects module versions that match, or are greater than, version 1.2.0. This form may be used if there is a dependency on a data node introduced in version 1.2.0. This is expected to be the most commonly used form of 'import by version'.

Includes versions: 1.2.0, 1.2.1M, 1.2.2M, 1.3.0, 1.3.1, 2.0.0, 3.0.0 and 3.1.0.

```
import example-module {  
    semver:version 1.2.0+;  
}
```

The next example selects module versions that match, or are greater than 1.1.0, but excluding all 1.1.x and 1.2.x 'M' versions. This form may be needed if structural non backwards compatible changes are introduced in a patch 'M' version. Generally, it is advisable to avoid making such changes.

Includes versions: 1.1.0, 1.1.1m, 1.2.0, 1.3.0, 1.3.1, 2.0.0, 3.0.0, and 3.1.0.

```
import example-module {  
    semver:version 1.1.0-1.1.1;  
    semver:version 1.2.0;  
    semver:version 1.3.0+;  
}
```

The last example selects all module versions with a major version number of 1. This form may be useful if significant non backwards compatible changes have been introduced in version 2.0.0 that break import backwards compatibility.

Includes versions: 1.0.0, 1.1.0, 1.1.1m, 1.1.2M, 1.2.0, 1.2.1M, 1.2.2M, 1.3.0 and 1.3.1.

```
import example-module {  
    semver:version 1.0.0-1.MAX.MAX;  
}
```

#### 4. Classifying changes in YANG modules

[RFC7950] chapter 11 defines the rules for what constitutes a backwards compatible change in YANG 1.1. However, the YANG semantic versioning scheme defined in this document uses a slightly modified version of this scheme, and also provides rules to classify changes as editorial, backwards-compatible, or non-backwards-compatible.

##### 4.1. Editorial changes

Any changes that do not change the ordering or meaning of the YANG module in any way are classified as 'editorial'. The following rules define 'editorial':

- o Changing any 'description' statement if it does not change the semantic meaning of the statement it relates to. E.g., fixing spelling or grammar, or changing layout, are all allowed.
- o Adding or updating 'reference' statements.
- o Adding or updating the 'organization' statement.
- o Adding a new 'revision' or 'semver:module-version' statement, or correcting a previous 'revision' or 'semver:module-version' statement.
- o A module may be split into a set of submodules or a submodule may be removed, provided the definitions in the module do not change except in the ways described above.

##### 4.2. Backwards-compatible changes

[RFC7950] chapter 11 defines the rules for what constitutes a backwards-compatible change in YANG 1.1. The document updates these rules in the following ways:

- o Adding or changing a 'status' node to 'obsolete' is not a backwards-compatible change. Other changes/additions of status elements are backwards-compatible, as per [RFC7950].

- o Changing the ordering of statements is allowed if it does not change the ordering of an rpc's 'input' substatements.

#### 4.3. Non-backwards-compatible changes

All other changes to YANG modules that are not classified as 'editorial' or 'backwards-compatible' are defined as being non-backwards-compatible.

Examples of non-backwards-compatible changes include:

- o Deleting a data node, or changing it to status obsolete.
- o Changing the name, type, or units of a data node.
- o Modifying the description in a way that changes the semantic meaning of the data node.
- o Any changes that change or reduce the allowed value set of the data node, either through changes in the type definition, or the addition or changes to 'must' statements, or changes in the description.
- o Adding or modifying 'when' statements that reduce when the data node is available in the schema.
- o Making the statement conditional on if-feature.

#### 5. Updates to ietf-yang-library

YANG library [RFC7895] [RFC8525] is modified to support semantic versioning in three ways.

##### 5.1. Advertising module version number

The ietf-semver YANG module augments the 'module' list in ietf-yang-library with a 'version' leaf to optionally declare the YANG semantic version of each module.

##### 5.2. Resolving ambiguous module imports

A YANG datastore schema, defined in [RFC8525], can specify multiple revisions of a YANG module in the schema using the 'import-only' list, with the requirement from [RFC7950] that only a single revision of a YANG module may be implemented.

If a YANG module import statement does not specify a specific version or revision within the datastore schema then it could be ambiguous as

to which module revision the import statement should resolve to. Hence, a datastore schema constructed by a client using the information contained in YANG library may not exactly match the datastore schema actually used by the server.

The following rules remove the ambiguity:

If a module import statement could resolve to more than one module revision defined in the datastore schema, and one of those revisions is implemented (i.e., not an 'import-only' module), then the import statement MUST resolve to the revision of the module that is defined as being implemented by the datastore schema.

If a module import statement could resolve to more than one module revision defined in the datastore schema, and none of those revisions are implemented, but one of more modules revisions specify a YANG semantic version, then the import MUST resolve to the module with the greatest version number, according to the version comparison rules in Section 3.

If a module import statement could resolve to more than one module revision defined in the datastore schema, none of those revisions are implemented, and none of the modules revisions have a YANG semantic version number, then the import MUST resolve to the module that has the most recent revision date.

### 5.3. Reporting how deprecated and obsolete nodes are handled

The ietf-semver YANG module augments YANG library with two leaves to allow a server to report how it handles status 'deprecated' and status 'obsolete' nodes. The leaves are:

**deprecated-nodes-implemented:** If present, this leaf indicates that all schema nodes with a status 'deprecated' child statement are implemented equivalently as if they had status 'current', or otherwise deviations MUST be used to explicitly remove 'deprecated' nodes from the schema. If this leaf is absent then the behavior is unspecified.

**obsolete-nodes-absent:** If present, this leaf indicates that the server does not implement any status 'obsolete' nodes. If this leaf is absent then the behaviour is unspecified.

Implementations that implement the YANG semantic versioning scheme defined in this document MUST set the 'deprecated-nodes-implemented' leaf because the refined module update rules in Section 4 require that this is how servers handle 'deprecated' and 'obsolete' nodes to comply with YANG module semantic versioning.

If a server does not set the 'deprecated-nodes-implemented' leaf, then clients MUST NOT rely solely on the YANG module semantic version number to determine whether two module versions are backwards compatible, and MUST also consider whether the status of any nodes has changed to 'deprecated' and whether those nodes are implemented by the server.

## 6. YANG status description extension

The ietf-semver module specifies the YANG extension 'status-description' that can be used as a substatement of the status statement. The argument to this extension can contain freeform text to help readers of the module understand why the node was deprecated or made obsolete, when it is anticipated that the node will no longer be available for use, and potentially reference other schema elements that can be used instead. An example is shown below.

```
leaf imperial-temperature {
  type int64;
  units "degrees Fahrenheit";
  status deprecated {
    semver:status-description
      "Imperial measurements are being phased out in favor
       of their metric equivalents. Use metric-temperature
       instead.";
  }
  description
    "Temperature in degrees Fahrenheit.";
}
```

## 7. Semantic versioning of YANG instance data

Instance data sets [I-D.ietf-netmod-yang-instance-file-format] do not have an associated YANG semantic version, as compatibility for instance data is undefined.

However, instance data may reference an associated YANG schema, and that schema could make use of semantic version numbers, both for the individual YANG modules that comprise the schema, and potentially for the entire schema itself (e.g., [I-D.rwilton-netmod-yang-packages]).

In this way, the versioning of a schema associated with an instance data set, may allow a client to determine whether the instance data could also be used in conjunction with other versions of the YANG schema, or other versions of the modules that define the schema.

One common scenario, where instance data may have to cope with changes to the schema is for the <startup> datastore when a server is

restarted with a different YANG schema (e.g. due to a software upgrade or downgrade). How a server restores the configuration from <startup> during such upgrades or downgrades is outside the scope of this specification.

## 8. Guidelines

### 8.1. Guidelines to YANG model authors

NBC changes to YANG models may cause problems to clients, who are consumers of YANG models, and SHOULD be avoided. However, there are cases where NBC changes are required, e.g. to fix an incorrect YANG model.

YANG model authors are recommended to minimize NBC changes and keep changes BC whenever possible.

The use of status "deprecated" with the status-description statement allows clients to plan a migration to alternative data nodes.

When NBC changes are introduced, consideration should be given to the impact on clients and YANG model authors SHOULD try to mitigate that impact.

#### 8.1.1. Use of YANG semantic versioning

Module authors should use the following guidance when applying the module version update rules specified in Section 2.3.

Updates to modules SHOULD be applied to the latest version of YANG modules, avoiding the use of the 'm|M' patch modifier. When used in this way, the YANG semantic version numbers are compatible with the versioning scheme defined by the semver.org 2.0.0 rules.

Changes to older versions of published YANG modules SHOULD be minimized, since there may be a greater impact on clients, and comparing between version numbers becomes more limited if the 'm|M' modifiers are used. However, if it is necessary to make such changes then the following guidelines apply:

Any changes SHOULD also be made to a new latest version of the YANG module, if appropriate.

Where possible, changes SHOULD be restricted to backwards-compatible changes only.



NBC changes MAY be made, subject to the constraints defined in Section 2.3. The impact to clients SHOULD be carefully considered and minimized if possible.

The version numbers associated with a module MUST never be reused. E.g., when updating module version 3.4.0 in a NBC manner the module author must verify whether version 4.0.0 is available for use and if that version was already used, the updated module must use version 3.4.1M instead.

Patch modifier letters (i.e. 'm' or 'M') are sticky. For example if version 3.4.1M is modified in a BC way, the next version is 3.4.2M. This is to indicate that 3.4.2M is not BC with 3.4.0, however it comes at the cost of not being able to indicate the type of change between 3.4.1M and 3.4.2M.

As explained in Appendix A.2.2, while programatically determining a semantic version is possible using tools (e.g. the pyang utility), human oversight is highly recommended because of some special cases which can not be detected by tools. Therefore, a model author SHOULD use both means to determine a model's semantic version.

#### 8.1.2. Making non-backwards-compatible changes to a YANG module

There are various valid situations where a YANG module has to be modified in a non-backwards-compatible way. Here are the different ways in which this can be done:

- o If the server can support NBC versions of the YANG module simultaneously using version selection, then the NBC changes MAY be done immediately. Clients would be required to select the version which they support and the NBC change would have no impact on them.
- o When possible, NBC changes are done incrementally to provide clients time to adapt to NBC changes.

Here are some guidelines on how non-backwards compatible changes can be made incrementally:

1. The changes should be made incrementally, e.g. a data node's status SHOULD NOT be changed directly from "current" to "obsolete" (see Section 4.7 of [RFC8407]), instead the status SHOULD first be marked "deprecated" and then when support is removed its status MUST be changed to "obsolete". Instead of using the "obsolete" status, the data node MAY be removed from the model but this has the risk of breaking modules which import the modified module.

2. A node with status "deprecated" MUST be supported for the solution described here to function properly.
3. A node with status "deprecated" SHOULD be available for at least one year before its status is changed to "obsolete", see Section 4.7 of [RFC8407].
4. Support for a node which is "obsolete" is indicated by the node "obsolete-nodes-present", see Section 5.
5. The new extension "status-description" SHOULD be used for nodes which are "obsolete" or "deprecated".
6. For status "deprecated", the "status-description" SHOULD also indicate until when support for the node is guaranteed. If there is a replacement data node, rpc, action or notification for the deprecated node, this SHOULD be stated in the "status-description".
7. When obsoleting or deprecating data nodes, the "deprecated" or "obsolete" status SHOULD be applied at the highest possible level in the data tree. For example, when deprecating all data nodes in a container, the "deprecated" status SHOULD be applied to the container. For clarity, the status MAY be added in all the affected nodes but the status-description SHOULD be added only at the highest level in the tree.

The following sections have examples on how non-backwards-compatible changes can be made.

#### 8.1.2.1. Removing a data node

Removing a leaf or container from the data tree, e.g. because support for the corresponding feature is being removed:

1. The node's status SHOULD be changed to "deprecated" and it MUST be supported for at least one year. This is a backwards-compatible change.
2. When the node is not available anymore, its status MUST be changed to "obsolete" and the "status-description" updated, this is a non-backwards-compatible change. The "status-description" SHOULD be used to explain why the node is not available anymore.

#### 8.1.2.2. Changing the type of a leaf node

Changing the type of a leaf-node. e.g. consider a "vpn-id" node of type integer being changed to a string:

1. The status of node "vpn-id" SHOULD be changed to "deprecated" and the node SHOULD be available for at least one year. This is a backwards-compatible change.
2. A new node, e.g. "vpn-name", of type string is added to the same location as the existing node "vpn-id". This new node has status "current" and its description SHOULD explain that it is replacing node "vpn-id".
3. During the period of time where both nodes are available, how the server behaves when either node is set is outside the scope of this document and will vary on a case by case basis. Here are some options:
  1. A server MAY prevent the new node from being set if the old node is already set (and vice-versa). The new node MAY have a when statement to achieve this. The old node MUST NOT have a when statement since this would be a non-backwards-compatible change, but the server MAY reject the old node from being set if the new node is already set.
  2. If the new node is set and a client does a get or get-config operation on the old node, the server MAY map the value. For example, if the new node "vpn-name" has value "123" then the server MAY return integer value 123 for the old node "vpn-id". However, if the value can not be mapped, we need a way of returning "unsupported" TBD.
4. When node "vpn-id" is not available anymore, its status MUST be changed to "obsolete" and the "status-description" is updated. This is a non-backwards-compatible change.

#### 8.1.2.3. Reducing the range of a leaf node

#### 8.1.2.4. Changing the key of a list

#### 8.1.2.5. Renaming a node

#### 8.1.2.6. Changing a default value

## 8.2. Guidelines to YANG model clients

Guidelines for clients of modules using YANG semantic versioning:

- o Clients SHOULD be liberal when processing data received from a server. For example, the server may have increased the range of an operational node causing the client to receive a value which is outside the range of the YANG model revision it was coded against
- o Clients SHOULD monitor changes to published YANG modules through their version numbers, and use appropriate tooling to understand the specific changes between module versions. In particular, clients SHOULD NOT migrate to NBC versions of a module without first understanding the specifics of the NBC changes.
- o Clients SHOULD plan to make changes to match published status changes. When a node's status changes from "current" to "deprecated", clients SHOULD plan to stop using that node in a timely fashion. When a node's status changes to "obsolete", clients MUST stop using that node.

## 9. Semantic Version Extension YANG Modules

YANG module with extensions for defining a module's YANG semantic version number, and importing by version.

```
<CODE BEGINS> file "ietf-semver@2019-02-07.yang"
module ietf-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-semver";
  prefix semver;

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:    <https://datatracker.ietf.org/wg/netmod/>
     WG List:   <mailto:netmod@ietf.org>

     Author:    Benoit Claise
                <mailto:bclaise@cisco.com>

     Author:    Joe Clarke
                <mailto:jclarke@cisco.com>

     Author:    Reshad Rahman
                <mailto:rrahman@cisco.com>

     Author:    Robert Wilton
```

```
<mailto:rwilton@cisco.com>

Author:   Kevin D'Souza
         <mailto:kd6913@att.com>

Author:   Balazs Lengyel
         <mailto:balazs.lengyel@ericsson.com>

Author:   Jason Sterne
         <mailto:jason.sterne@nokia.com>";
description
  "This module contains a definition for a YANG 1.1 extension to
  express the semantic version of YANG modules.";

revision 2019-02-27 {
  description
    "* Move YANG library augmentations into a separate module.
    * Update references.";
  reference
    "draft-verdt-netmod-yang-semver:
    YANG Semantic Versioning for Modules";
  semver:module-version "0.3.0";
}

revision 2018-04-05 {
  description
    "* Properly import ietf-yang-library.
    * Fix the name of module-semver => module-version.
    * Fix regular expression syntax.
    * Augment yang-library with booleans as to whether or not
      deprecated and obsolete nodes are present.
    * Add an extension to enable import by semantic version.
    * Add an extension status-description to track deprecated
      and obsolete reasons.
    * Fix yang-library augments to use 7895bis.";
  reference
    "draft-clacla-netmod-yang-model-update:
    New YANG Module Update Procedure";
  semver:module-version "0.2.1";
}
revision 2017-12-15 {
  description
    "Initial revision.";
  reference
    "draft-clacla-netmod-yang-model-update:
    New YANG Module Update Procedure";
  semver:module-version "0.1.1";
}
```

```
typedef version {
  type string {
    pattern '[0-9]{1,5}\.[0-9]{1,5}\.[0-9]{1,5}(m|M)?';
  }
  description
    "The type used to represent a YANG semantic version number.

    The YANG semver version number is expressed as a string of the
    form: 'X.Y.Zv'; where X, Y, and Z each represent non-negative
    integers smaller than 32768, and v represents an optional
    single character suffix: 'm' or 'M'.

    o 'X' is the MAJOR version.  Changes in the major version
      number indicate changes that are non-backwards-compatible to
      versions with a lower major version number.

    o 'Y' is the MINOR version.  Changes in the minor version
      number indicate changes that are backwards-compatible to
      versions with the same major version number, but a lower
      minor version number.

    o 'Zv' is the PATCH version and modifier.  Changes in the patch
      version number can indicate editorial, backwards-compatible,
      or non-backwards-compatible changes relative to versions with
      the same major and minor version numbers, but lower patch
      version number, depending on what form modifier 'v' takes:

      * 'M' - the change represents a non-backwards-compatible
        change

      * 'm' - the change represents a backwards-compatible change

      * If the modifier letter is absent, the change represents an
        editorial change";

  reference
    "draft-verdt-netmod-yang-semver: YANG Semantic Versioning";
}

extension module-version {
  argument semver;
  description
    "The version number for the module revision it is used in.

    This format of the argument matches the type version.

    The rules for updating the module-version number are described
    in section XXX of 'YANG Semantic Versioning for Modules';
```

By comparing the module-version between two revisions of a given module, one can determine if different revisions are backwards compatible or not, as well as whether or not new features have been added to a newer revision.

If a module contains this extension it indicates that for this module the updated status and update rules as this described in RFC XXXX are used.

The statement MUST only be a substatement of the 'revision' statements. Zero or one module-version statement is allowed per parent statement. No substatements are allowed.

'revision' statements in submodules MAY contain a 'module-version' statement for documentation purposes, but its meaning is undefined, and has no effect on the including module's semantic version."

reference

```
"draft-verdt-netmod-yang-semver:
  YANG Semantic Versioning for Modules";
```

```
}
```

```
extension import-versions {
  argument version-clause;
  description
```

```
"This extension specifies an acceptable set of semantic
  versions of a given module that may be imported.
```

The statement MUST only be a substatement of the import statement.

The statement MUST NOT be present if the import has a revision-date substatement.

The statement MUST NOT be present if the imported module does not support semantic versioning.

Zero or more versions statements are allowed per parent statement. No substatements are allowed.

The version-clause argument MUST follow one of the below patterns:

```
(i)  "+" \d+\.\d+\.\d+ '+'
```

Matches exact version, e.g. 3.6.1

```
(ii) "+" '\d+\.\d+\.\d+\+ '+'
```

Matches exact version or greater, e.g. 3.6.1+

```
(iii) "+" \d+\.\d+\.\d+[-\d+\.\d+|MAX)\.(\d+|MAX)+ '+'
Matches inclusive range,
e.g. 3.6.1-7.8.4, or 3.2.1-3.MAX.MAX";
```

```
reference
  "draft-verdt-netmod-yang-semver: Import by Semantic Version";
}

extension status-description {
  argument description;
  description
    "Freeform text that describes why a given node has been
    deprecated or made obsolete. This may point to other schema
    elements that can be used in lieu of the given node.

    This statement MUST only be used as a substatement of the
    status statement

    Zero or more status-description statements are allowed per
    parent statement. No substatements are allowed.";
  reference
    "draft-verdt-netmod-yang-semver: YANG status description
    extension";
}
}
<CODE ENDS>
```

YANG module with augmentations to YANG Library to support semantic version numbers.

```
<CODE BEGINS> file "ietf-yl-semver@2019-02-07.yang"
module ietf-yl-semver {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yl-semver";
  prefix yl-semver;

  import ietf-semver {
    prefix semver;
  }

  import ietf-yang-library {
    prefix yanglib;
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/netmod/>
```



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

Author:   Benoit Claise
          <mailto:bclaise@cisco.com>

Author:   Joe Clarke
          <mailto:jclarke@cisco.com>

Author:   Reshad Rahman
          <mailto:rrahman@cisco.com>

Author:   Robert Wilton
          <mailto:rwilton@cisco.com>

Author:   Kevin D'Souza
          <mailto:kd6913@att.com>

Author:   Balazs Lengyel
          <mailto:balazs.lengyel@ericsson.com>

Author:   Jason Sterne
          <mailto:jason.sterne@nokia.com>";
description
  "This module contains augmentations to YANG Library to add module
  level semantic version numbers and to provide an indication of
  how deprecated and obsolete nodes are handled by the server.";

semver:module-version "0.1.0";

revision 2019-02-27 {
  description
    "Moved YANG library augmentations into a separate module.";
  reference
    "draft-verdt-netmod-yang-semver:
    YANG Semantic Versioning for Modules";
  semver:module-version "0.1.0";
}

augment "/yanglib:yang-library/yanglib:module-set/yanglib:module" {
  description
    "Augmentation modules with a semantic version.";
  leaf version {
    type semver:version;
    description
      "The semantic version for this module. The version MUST
      match the semver:version value in specific revision of the
      module loaded in this module-set.";
    reference
```

```
        "draft-verdt-netmod-yang-semver: YANG Semantic Versioning";
    }
}

augment "/yanglib:yang-library/yanglib:schema" {
    description
        "Augmentations to the ietf-yang-library module to indicate how
        deprecated and obsoleted nodes are handled for each datastore
        schema supported by the server.";

    leaf deprecated-nodes-implemented {
        type empty;
        description
            "If present, this leaf indicates that all schema nodes with a
            status 'deprecated' child statement are implemented
            equivalently as if they had status 'current', or otherwise
            deviations MUST be used to explicitly remove 'deprecated'
            nodes from the schema. If this leaf is absent then the
            behavior is unspecified.";
        reference
            "draft-verdt-netmod-yang-semver: Reporting how deprecated and
            obsolete nodes are handled";
    }
    leaf obsolete-nodes-absent {
        type empty;
        description
            "If present, this leaf indicates that the server does not
            implement any status 'obsolete' nodes. If this leaf is
            absent then the behaviour is unspecified.";
        reference
            "draft-verdt-netmod-yang-semver: Reporting how deprecated and
            obsolete nodes are handled";
    }
}
}
}
<CODE ENDS>
```

## 10. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The design team consists of the following members whom have worked on the YANG versioning project:

- o Balazs Lengyel
- o Benoit Claise
- o Ebben Aries

- o Jason Sterne
- o Joe Clarke
- o Juergen Schoenwaelder
- o Mahesh Jethanandani
- o Michael (Wangzitao)
- o Qin Wu
- o Reshad Rahman
- o Rob Wilton

The initial revision of this document was refactored and built upon [I-D.claccla-netmod-yang-model-update].

Discussions on the use of Semver for YANG versioning has been held with authors of the OpenConfig YANG models. We would like thank both Anees Shaikh and Rob Shakir for their input into this problem space.

## 11. Security Considerations

The document does not define any new protocol or data model. There are no security impacts.

## 12. IANA Considerations

### 12.1. YANG Module Registrations

The following YANG module is requested to be registered in the "IANA Module Names" registry:

The ietf-semver module:

Name: ietf-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-semver

Prefix: semver

Reference: [RFCXXXX]

The ietf-yl-semver module:

Name: ietf-yl-semver

XML Namespace: urn:ietf:params:xml:ns:yang:ietf-yl-semver

Prefix: yl-semver

Reference: [RFCXXXX]

## 13. References

### 13.1. Normative References

- [I-D.verdt-netmod-yang-versioning-reqs]  
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.
- [RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <<https://www.rfc-editor.org/info/rfc7895>>.
- [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>>.
- [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>>.
- [RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <<https://www.rfc-editor.org/info/rfc8525>>.

### 13.2. Informative References

- [I-D.clacla-netmod-model-catalog]  
Clarke, J. and B. Claise, "YANG module for yangcatalog.org", draft-clacla-netmod-model-catalog-03 (work in progress), April 2018.
- [I-D.clacla-netmod-yang-model-update]  
Claise, B., Clarke, J., Lengyel, B., and K. D'Souza, "New YANG Module Update Procedure", draft-clacla-netmod-yang-model-update-06 (work in progress), July 2018.

- [I-D.claise-semver]  
Claise, B., Barnes, R., and J. Clarke, "Semantic Versioning and Structure for IETF Specifications", draft-claise-semver-02 (work in progress), January 2018.
- [I-D.ietf-netmod-yang-instance-file-format]  
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-02 (work in progress), February 2019.
- [I-D.openconfig-netmod-model-catalog]  
Shaikh, A., Shakir, R., and K. D'Souza, "Catalog and registry for YANG models", draft-openconfig-netmod-model-catalog-02 (work in progress), March 2017.
- [I-D.rwilton-netmod-yang-packages]  
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-00 (work in progress), December 2018.
- [openconfigsemver]  
"Semantic Versioning for Openconfig Models",  
<<http://www.openconfig.net/docs/semver/>>.
- [semver] "Semantic Versioning 2.0.0", <<https://www.semver.org>>.
- [yangcatalog]  
"YANG Catalog", <<https://yangcatalog.org>>.

### 13.3. URIs

- [1] <https://github.com/netmod-wg/yang-ver-dt/issues/14>
- [2] <https://github.com/netmod-wg/yang-ver-dt/issues/11>
- [3] <https://github.com/netmod-wg/yang-ver-dt/issues/13>
- [4] <https://github.com/netmod-wg/yang-ver-dt/issues/12>
- [5] <https://github.com/netmod-wg/yang-ver-dt/issues/10>
- [6] <https://github.com/netmod-wg/yang-ver-dt/issues/9>
- [7] <https://github.com/netmod-wg/yang-ver-dt/issues/8>
- [8] <https://github.com/netmod-wg/yang-ver-dt/issues/7>
- [9] <https://github.com/netmod-wg/yang-ver-dt/issues/6>

- [10] <https://github.com/netmod-wg/yang-ver-dt/issues/5>
- [11] <https://github.com/netmod-wg/yang-ver-dt/issues/4>
- [12] <https://github.com/netmod-wg/yang-ver-dt/issues/15>
- [13] <https://github.com/netmod-wg/yang-ver-dt/issues/2>

## Appendix A. Appendix

### A.1. Open Issues

Open issues are being tracked at <<https://github.com/netmod-wg/yang-ver-dt/issues>>. Currently open issues are:

- o Do we need a new version of YANG? #14 [1]
- o Add guidance text about warning NBC changes might break imports #11 [2]
- o Add a naming convention for versioned YANG file#13 [3]
- o Define editorial, bc, nbc impact of adding, changing, removing extension stmts#12 [4]
- o How to version modules in IETF drafts (after they have been published at 1.0.0 or later#10 [5]
- o The solution does not strictly support semver 2.0.0#9 [6]
- o Are whitespace changes allow between two module instances with the same version (or revision)?#8 [7]
- o Do we assume that a module has an implicit semver if none as been specified?#7 [8]
- o Is changing the ordering of nodes an NBC change?#6 [9]
- o Should version statement be at top level or under revision statement?#5 [10]
- o Figure out whether changing the imports constitute a BC or NBC change#4 [11]
- o Does BC or NBC depend on whether the node is config true/false?#15 [12]
- o Status obsolete nodes#2 [13]

## A.2. Derived Semantic Version

This temporary text is intended to be moved to a separate draft that describes the tool based approach for versioning YANG modules mentioned in Section 1.2.

### A.2.1. The Derived Semantic Version

If an explicitly defined semantic version is not available in the YANG module, it is possible to algorithmically calculate a derived semantic version. This can be used for modules not containing a definitive semantic-version as defined in this document or as a starting value when specifying the definitive semantic-version. Be aware that this algorithm may sometimes incorrectly classify changes between the categories non-compatible, compatible or error-correction.

### A.2.2. Implementation Experience

[yangcatalog] uses the pyang utility to calculate the derived-semantic-version for all of the modules contained within the catalog. [yangcatalog] contains many revisions of the same module in order to provide its derived-semantic-version for module consumers to know what has changed between revisions of the same module.

Two distinct leafs in the YANG module

[I-D.clacla-netmod-model-catalog] contain this semver notation:

- o the semantic-version leaf contains the value embedded within a YANG module (if it is available).
- o the derived-semantic-version leaf is established by examining the the YANG module themselves. As such derived-semantic-version only takes syntax into account as opposed to the meaning of various elements when it computes the semantic version.
- o The algorithm used to produce the derived-semantic-version is as follows:
  1. Order all modules of the same name by revision from oldest to newest. Include module revisions that are not available, but which are defined in the revision statements in one of the available module versions.
  2. If module A, revision N+1 has failed compilation, bump its derived semantic MAJOR version. For unavailable module versions assume non-backward compatible changes were done., thus bump its derived semantic MAJOR version.

3. Else, run "pyang --check-update-from" on module A, revision N and revision N+1 to see if backward-incompatible changes exist.
4. If backward-incompatible changes exist, bump module A, revision N+1's derived MAJOR semantic version.
5. If no backward-incompatible changes exist, compare the pyang trees of module A, revision N and revision N+1.
6. If there are structural differences (e.g., new nodes), bump module A, revision N+1's derived MINOR semantic version.
7. If no structural differences exist, bump module A, revision N+1's derived PATCH semantic version.

The pyang utility checks many of the points listed in section 11 of [RFC7950] for known module incompatibilities. While this approach is a good way to programmatically obtain a semantic version number, it does not address all cases whereby a major version number might need to be increased. For example, a node may have the same name and same type, but its meaning may change from one revision of a module to another. This represents a semantic change that breaks backward compatibility, but the above algorithm would not find it. Therefore, additional, sometimes manual, rigor must be done to ensure a proper version is chosen for a given module revision.

#### Authors' Addresses

Benoit Claise  
Cisco Systems, Inc.  
De Kleetlaan 6a b1  
1831 Diegem  
Belgium  
  
Phone: +32 2 704 5622  
Email: bclaise@cisco.com

Joe Clarke  
Cisco Systems, Inc.  
7200-12 Kit Creek Rd  
Research Triangle Park, North Carolina  
United States of America  
  
Phone: +1-919-392-2867  
Email: jclarke@cisco.com



Reshad Rahman  
Cisco Systems, Inc.

Email: rrahman@cisco.com

Robert Wilton (editor)  
Cisco Systems, Inc.

Email: rwilton@cisco.com

Balazs Lengyel  
Ericsson  
Magyar Tudosok Korutja  
1117 Budapest  
Hungary

Phone: +36-70-330-7909  
Email: balazs.lengyel@ericsson.com

Jason Sterne  
Nokia

Email: jason.sterne@nokia.com

Kevin D'Souza  
AT&T  
200 S. Laurel Ave  
Middletown, NJ  
United States of America

Email: kd6913@att.com

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: May 27, 2019

J. Clarke, Ed.  
Cisco Systems, Inc.  
November 23, 2018

YANG Module Versioning Requirements  
draft-verdt-netmod-yang-versioning-reqs-02

Abstract

This document describes the problems that can arise because of the YANG language module update rules, that require all updates to YANG module preserve strict backwards compatibility. It also defines the requirements on any solution designed to solve the stated problems. This document does not consider possible solutions, nor endorse any particular solution.

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 May 27, 2019.

Copyright Notice

Copyright (c) 2018 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 Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Background . . . . .	2
2.1. Striving for model perfection . . . . .	3
2.2. Some YANG Modules Are Not Backwards-Compatible . . . . .	3
2.3. Non-Backwards-Compatible Errors . . . . .	4
2.4. No way to easily decide whether a change is Backwards-Compatible . . . . .	4
2.5. No good way to specify which module revision to import . . . . .	5
2.6. Early Warning about Removal . . . . .	6
2.7. Clear Indication of Node Support . . . . .	6
3. Terminology and Conventions . . . . .	7
4. The Problem Statement . . . . .	7
5. Requirements of a YANG Versioning Solution . . . . .	9
6. Contributors . . . . .	11
7. Acknowledgments . . . . .	11
8. Security Considerations . . . . .	11
9. IANA Considerations . . . . .	12
10. References . . . . .	12
10.1. Normative References . . . . .	12
10.2. Informative References . . . . .	12
Author's Address . . . . .	12

## 1. Introduction

This requirements document initially considers some of the existing YANG module update rules, then describes the problems that arise due to those rules embracing strict backwards compatibility, and finally defines requirements on any solution that may be designed to solve these problems by providing an alternative YANG versioning strategy.

## 2. Background

The YANG data modeling language [RFC7950] specifies strict rules for updating YANG modules (see section 11 "Updating a Module"). Citing a few of the relevant rules:

1. "As experience is gained with a module, it may be desirable to revise that module. However, changes to published modules are not allowed if they have any potential to cause interoperability problems between a client using an original specification and a server using an updated specification."

2. "Note that definitions contained in a module are available to be imported by any other module and are referenced in "import" statements via the module name. Thus, a module name MUST NOT be changed. Furthermore, the "namespace" statement MUST NOT be changed, since all XML elements are qualified by the namespace."
3. "Otherwise, if the semantics of any previous definition are changed (i.e., if a non-editorial change is made to any definition other than those specifically allowed above), then this MUST be achieved by a new definition with a new identifier."
4. "deprecated indicates an obsolete definition, but it permits new/continued implementation in order to foster interoperability with older/existing implementations."

The rules described above, along with other similar rules, causes various problems, as described in the following sections:

#### 2.1. Striving for model perfection

The points made above lead to the logical conclusion that the standardized YANG modules have to be perfect on day one (at least the structure and meaning), which in turn might explain why IETF YANG modules take so long to standardize. Shooting for perfection is obviously a noble goal, but if the perfect standard comes too late, it doesn't help the industry.

#### 2.2. Some YANG Modules Are Not Backwards-Compatible

As we learn from our mistakes, we're going to face more and more non-backwards-compatible YANG modules. An example is the YANG data model for L3VPN service delivery [RFC8049], which, based on implementation experience, has been updated in a non-backwards-compatible way by [RFC8299].

While Standards Development Organization (SDO) YANG modules are obviously better for the industry, we must recognize that many YANG modules are actually generated YANG modules (for example, from internal databases), which is sometimes the case for vendor modules [RFC8199]. From time to time, the new YANG modules are not backwards-compatible.

Old module parts that are no longer needed, no longer supported, or are not used by consumers need to be removed from modules. It is often hard to decide which parts are no longer needed/used; still the need and practice of removing old parts exist. While it is rare in standard modules it is more common in vendor YANG modules where the usage of modules is more controlled.

The problems described in Section 2.7 may also result in incompatible changes.

In such cases, it would be better to indicate how backwards-compatible a given YANG module actually is.

As modules are sometimes updated in an incompatible way the current assumption that once a YANG module is defined all further revisions can be freely used as they are compatible is not valid.

### 2.3. Non-Backwards-Compatible Errors

Sometimes small errors force us to make non-backwards-compatible updates. As an example imagine that we have a string with a complex pattern (e.g., an IP address). Let's assume the initial pattern incorrectly allows IP addresses to start with 355. In the next version this is corrected to disallow addresses starting with 355. Formally this is a non-backwards-compatible change as the value space of the string is decreased. In reality an IP address and the implementation behind it was never capable of handling an address starting with 355. So practically this is a backwards-compatible change, just like a correction of the description statement. Current YANG rules are ambiguous as to whether non-backwards-compatible bug fixes are allowed without also requiring a module name change.

### 2.4. No way to easily decide whether a change is Backwards-Compatible

A management system, SDN controller, or any other user of a module should be capable of easily determining the compatibility between two module versions. Higher level logic for a network function, something that cannot be implemented in a purely model driven way, is always dependent on a specific version of the module. If the client finds that the module has been updated on the network node, it has to decide if it tries to handle it as it handled the previous version of the model or if it just stops, to avoid problems. To make this decision the client needs to know if the module was updated in a backwards-compatible way or not.

This is not possible to decide today because of the following:

- o It is sometimes necessary to change the semantic behavior of a data node, action or rpc while the YANG definition does not change (with the possible exception of the description statement). In such a case it is impossible to determine whether the change is backwards-compatible just by looking at the YANG statements. It's only the human model designer who can decide.

- o Problems with the deprecated and obsolete status statement, Section 2.7
- o YANG module authors might decide to violate YANG 1.1 update rules for some of the reasons above.

Finding status changes or violations of update rules need a line-by-line comparison of the old and new modules is a tedious task.

## 2.5. No good way to specify which module revision to import

If a module (MOD-A) is imported by another one (MOD-B) the importer may specify which revision must be imported. Even if MOD-A is updated in a backwards-compatible way not all revisions will be suitable, e.g., a new MOD-B might need the newest MOD-A. However, both specifying or omitting the revision date for import leads to problems.

If the import by revision-date is specified

- o If corrections are made to MOD-A these would not have any effect as the import's revision date would still point to the uncorrected earlier YANG module revision.
- o If MOD-A is updated in a backwards-compatible way because another importer (MOD-C) needs some functionality, the new MOD-A could be used by MOD-B, but specifying the exact import revision-date prevents this. This will force the implementers to import two different revisions of MOD-A, forcing them to maintain old MOD-A revisions unnecessarily.
- o If multiple modules import different revisions of MOD-A the human user will need to understand the subtle differences between the different revisions. Small differences would easily lead to operator mistakes as the operator will rarely check the documentation.
- o Tooling/SW is often not prepared to handle multiple revisions of the same YANG module.

If the import revision-date is not specified

- o any revision of MOD-A may be used including unsuitable ones. Older revisions may be lacking functionality MOD-B needs. Newer MOD-A revisions may obsolete definitions used by MOD-B in which case these must not be used by MOD-B anymore.

- o As it is not specified which revisions of MOD-A are suitable for MOD-B. The problem has to be solved on a case by case basis studying all the details of MOD-A and MOD-B which is considerable work.

## 2.6. Early Warning about Removal

If a schema part is considered old/bad we need to be able to give advance warning that it will be removed. As this is an advance warning the part must still be present and usable in the current revision; however, it will be removed in one of the next revisions. The deprecated statement cannot be reliably used for this purpose both because deprecated nodes may not be implemented and also there is no mandate that text be provided explaining the deprecation.

We need the advance warning to allow users of the module time to plan/execute migration away from the deprecated functionality. Deprecation should be accompanied by information whether the functionality will just disappear or that there is an alternative, possibly more advanced solution that should be used.

Vendors use such warnings often, but the NMDA related redesign of IETF modules is also an example where it would be useful for IETF. As another example, see the usage of deprecated in the Java programming language.

## 2.7. Clear Indication of Node Support

The current definition of deprecated and obsolete in [RFC7950] (as quoted below) is problematic and should be corrected.

- o "deprecated" indicates an obsolete definition, but it permits new/continued implementation in order to foster interoperability with older/existing implementations.
- o "obsolete" means that the definition is obsolete and SHOULD NOT be implemented and/or can be removed from implementations.

YANG is considered an interface contract between the server and the client. The current definitions of deprecated and obsolete mean that a schema node that is either deprecated or obsolete may or may not be implemented. The client has no way to find out which is the case except for by trying to write or read data at the leaf in question. This probing would need to be done for each separate data-node, which is not a trivial thing to do. This "may or may not" is unacceptable in a contract. In effect, this works as if there would be an if-feature statement on each deprecated schema node where the server

does not advertise whether the feature is supported or not. Why is it not advertised?

### 3. Terminology and Conventions

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

In addition, this document uses the following terminology:

- o YANG module revision: An instance of a YANG module, with no implied ordering or backwards compatibility between different revisions of the same module."
- o YANG module version: A YANG module revision, but also with an implied partial ordering relationship between other versions of the same module. Each module version must be uniquely identifiable.
- o Non-backwards-compatible (NBC): In the context of this document, the term 'non-backwards-compatible' refers to a change or set of changes between two YANG module revisions that do not adhere to the list of allowable changes specified in Section 11 "Updating a Module" of [RFC7950], with the following additional clarification:
  - \* Any addition of, or change to, a "status" statement that allows a server to remove support for a schema node is considered a non-backwards-compatible change

### 4. The Problem Statement

Considering the issues described in the background, the problem definition can be summarized as follows.

Development of data models for a large collection of communication protocols and system components is difficult and typically only manageable with an iterative development process. Agile development approaches advocate evolutionary development, early delivery, and continual improvement. They are designed to support rapid and flexible response to change. Agile development has been found to be very successful in a world where the objects being modeled undergo constant changes.

The current module versioning scheme relies on the fundamental idea that a definition, once published, never changes its semantics. As a consequence, if a new definition is needed with different non-backwards-compatible semantics, then a new definition must be created



to replace the old definition. The advantage of this versioning scheme is that a definition identified by a module name and a path has fixed semantics that never change. (The details are a bit more nuanced but we simplify things here a bit in order to get the problems worked out clearly.)

There are two main disadvantages of the current YANG versioning scheme:

- o Any non-backwards-compatible change of a definition requires either a new module name or a new path. This has been found costly to support in implementations, in particular on the client side.
- o Since non-backwards-compatible changes require either a new module name or a new path, such changes will impact other modules that import definitions. In fact, with the current module versioning scheme other modules have to opt-in in order to use the new version. This essentially leads to a ripple effect where a non-backwards-compatible change of a core module causes updates on a potentially large number of dependent modules.

Other problems experienced with the current YANG versioning scheme are the following:

- o YANG has a mechanism to mark definitions deprecated but it leaves it open whether implementations are expected to implement deprecated definitions and there is no way (other than trial and error) for a client to find out whether deprecated definitions are supported by a given implementation.
- o YANG does not have a robust mechanism to document which data definitions have changed and to provide guidance how implementations should deal with the change. While it is possible to have this described in general description statements, having these details embedded in general description statements does not make this information accessible to tools.
- o YANG data models often do not exist in isolation and they interact with other software systems or data models that often do allow (controlled) non-backwards-compatible changes. In some cases, YANG models are mechanically derived from other data models that do allow (controlled) non-backwards-compatible changes. In such situations, a robust mapping to YANG requires to have version numbers exposed as part of the module name or a path definition, which has been found to be expensive on the client side (see above).

Given the need to support agile development processes and the disadvantages and problems of the current YANG versioning scheme described above, it is necessary to develop requirements and solutions for a future YANG versioning scheme that better supports agile development processes, whilst retaining the ability for servers to handle clients using older versions of YANG modules.

## 5. Requirements of a YANG Versioning Solution

The following is a list of requirements that a solution to the problems mentioned above MUST or SHOULD have. The list is grouped by similar requirements but is not presented in a set priority order.

1. Requirements related to making non-backwards-compatible updates to modules:
  - 1.1 A mechanism is REQUIRED to update a module in a non-backwards-compatible way without forcing all modules with import dependencies on the updated module from being updated at the same time (e.g. to change its import to use a new module name).
  - 1.2 Non-backwards-compatible updates of a module MUST not impact clients that only access data nodes of the module that have either not been updated or have been updated in backwards-compatible ways.
  - 1.3 A refined form of YANG's 'import' statement MUST be provided that is more restrictive than "import any revision" and less restrictive than "import a specific revision". Once non-backwards-compatible changes to modules are allowed, the refined import statement is used to express the correct dependency between modules.
  - 1.4 The solution MUST allow for backwards-compatible enhancements and bug fixes, as well as non-backwards-compatible bug fixes in non-latest-release modules.
2. Requirements related to identifying changes between different module revisions:
  - 2.1 Readers of modules, and tools that use modules, MUST be able to determine whether changes between two revisions of a module constitute a backwards-compatible or non-backwards-compatible version change. In addition, it MAY be helpful to identify whether changes represent bug fixes, new functionality, or both.

- 2.2 A mechanism SHOULD be defined to determine whether data nodes between two arbitrary YANG module revisions have (i) not changed, (ii) changed in a backwards-compatible way, (iii) changed in a non-backwards-compatible way.
3. Requirements related to supporting existing clients in a backwards-compatible way:
  - 3.1 The solution MUST provide a mechanism to allow servers to support existing clients in a backwards-compatible way.
  - 3.2 The solution MUST provide a mechanism to support clients that expect an older version of a given module when the current version has had non-backwards-compatible changes.
  - 3.3 Clients are expected to be able to handle unexpected instance data resulting from backwards-compatible changes.
4. Requirements related to managing and documenting the life cycle of data nodes:
  - 4.1 A mechanism is REQUIRED to allow a client to determine whether deprecated nodes are implemented by the server.
  - 4.2 If a data node is deprecated or obsolete then it MUST be possible to document in the YANG module what alternatives exist, the reason for the status change, or any other status related information.
  - 4.3 A mechanism is REQUIRED to indicate that certain definitions in a YANG module will become status obsolete in future revisions but definitions marked as such MUST still be implemented by compliant servers.
5. Requirements related to documentation and education:
  - 5.1 The solution MUST provide guidance to model authors and clients on how to use the new YANG versioning scheme.
  - 5.2 The solution is REQUIRED to describe how to transition from the existing YANG 1.0/1.1 versioning scheme to the new scheme.
  - 5.3 The solution MUST describe how the versioning scheme affects the interpretation of instance data and references to instance data, for which the schema definition has been updated in a non-backwards-compatible way.

## 6. Contributors

This document grew out of the YANG module versioning design team that started after IETF 101. The following people are members of that design team and have contributed to defining the problem and specifying the requirements:

- o Balazs Lengyel
- o Benoit Claise
- o Ebben Aries
- o Jason Sterne
- o Joe Clarke
- o Juergen Schoenwaelder
- o Mahesh Jethanandani
- o Michael (Wangzitao)
- o Qin Wu
- o Reshad Rahman
- o Rob Wilton

## 7. Acknowledgments

The design team would like to thank Christian Hopps and Vladimir Vassilev for their feedback and perspectives in shaping and fine tuning the versioning requirements.

One of the inspirations for solving the YANG module versioning comes from OpenConfig. The authors would like to thank Anees Shaikh and Rob Shakir for their helpful input.

## 8. Security Considerations

The document does not define any new protocol or data model. There is no security impact.

## 9. IANA Considerations

None

## 10. References

### 10.1. Normative References

- [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>>.
- [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>>.

### 10.2. Informative References

- [RFC8049] Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8049, DOI 10.17487/RFC8049, February 2017, <<https://www.rfc-editor.org/info/rfc8049>>.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module Classification", RFC 8199, DOI 10.17487/RFC8199, July 2017, <<https://www.rfc-editor.org/info/rfc8199>>.
- [RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <<https://www.rfc-editor.org/info/rfc8299>>.

### Author's Address

Joe Clarke (editor)  
Cisco Systems, Inc.  
7200-12 Kit Creek Rd  
Research Triangle Park, North Carolina  
United States of America

Phone: +1-919-392-2867  
Email: [jclarke@cisco.com](mailto:jclarke@cisco.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 12, 2019

R. Wilton  
R. Rahman  
Cisco Systems, Inc.  
March 11, 2019

YANG Schema Version Selection  
draft-wilton-netmod-yang-ver-selection-00

Abstract

This document defines protocol mechanisms to allow clients to choose which YANG schema to use for interactions with a server, out of the available YANG schema supported by a server. The provided functionality allow servers to support clients in a backwards compatible way, at the same time allowing for non-backwards-compatible updates to YANG modules.

This draft provides a solution to YANG versioning requirements 3.1 and 3.2.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Terminology and Conventions . . . . .	2
2. Introduction . . . . .	3
3. Background . . . . .	4
4. Objectives . . . . .	4
5. Solution Overview . . . . .	5
6. Version selection from a server perspective . . . . .	6
7. Version selection from a clients perspective . . . . .	7
8. Limitations of the solution . . . . .	7
9. Schema Version Selection YANG module . . . . .	8
10. YANG Module . . . . .	9
11. Security Considerations . . . . .	13
12. IANA Considerations . . . . .	14
13. Open Questions/Issues . . . . .	14
14. Acknowledgements . . . . .	14
15. References . . . . .	14
15.1. Normative References . . . . .	14
15.2. Informative References . . . . .	15
Authors' Addresses . . . . .	16

## 1. Terminology and Conventions

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.

This document uses terminology introduced in the YANG versioning requirements draft [I-D.verdt-netmod-yang-versioning-reqs].

This document also makes of the following terminology introduced in the Network Management Datastore Architecture [RFC8342]:

- o datastore schema

In addition, this document makes use of the following terminology:

- o bc: Used as an abbreviation for a backwards-compatible change.
- o nbc: Used as an abbreviation for a non-backwards-compatible change.

- o editorial change: A backwards-compatible change that does not change the YANG module semantics in any way.
- o YANG schema: The combined set of schema nodes for a set of YANG module revisions, taking into consideration any deviations and enabled features.
- o versioned schema: A YANG schema with an associated YANG semantic version number, e.g., as might be described by a YANG package.
- o schema set: A set of related versioned YANG schema, one for each datastore that is supported.

TODO - the bc/nbc/editorial terminology should probably be defined and referenced from the YANG module versioning solution draft. 'schema' and 'versioned schema' could be defined in the packages draft.

## 2. Introduction

This document describes how NETCONF and RESTCONF clients can choose a particular YANG schema they wish to choose to interact with a server with.

[I-D.verdt-netmod-yang-versioning-reqs] defines requirements that any solution to YANG versioning must have.

[I-D.verdt-netmod-yang-semver] specifies a partial solution to the YANG versioning requirements that focuses on using semantic versioning within individual YANG modules, but does not address all the requirements listed in the requirements draft. Of particular relevance here, requirements 3.1 and 3.2 are not addressed.

[I-D.rwilton-netmod-yang-packages] describes how sets of related YANG modules can be grouped together into a logical entity that is versioned using the YANG semantic versioning number scheme. Different packages can be defined for different sets of YANG modules, e.g., packages could be defined for the IETF YANG modules, OpenConfig YANG modules, a vendor's YANG modules. Different versions of these package definitions can be defined as the contents of these packages evolve over time, and as the versions of the YANG modules included in the package evolve.

This draft defines how YANG packages can be used to represent versioned datastore schema, and how clients can choose which versioned schemas to use during interactions with a device.



### 3. Background

There are three ways that the lifecycle of a data model can be managed:

1. Disallow all non-backwards-compatible updates to a YANG module. Broadly this is the approach adopted by [RFC7950], but it has been shown to be too inflexible in some cases. E.g. it makes it hard to fix bugs in a clean fashion - it is not clear that allowing two independent data nodes (one deprecated, one current) to configure the same underlying property is robustly backwards compatible in all scenarios, particularly if the value space and/or default values differ between the module revisions.
2. Allow non-backwards-compatible updates to YANG modules, and use a mechanism such as semantic version numbers to communicate the likely impact of any changes to module users, but require that clients handle non-backwards-compatible changes in servers by migrating to new versions of the modules. Without version selection, this is what the [I-D.verdt-netmod-yang-semver] approach likely achieves.
3. Allow non-backwards-compatible updates to YANG modules, but also provide mechanisms to allow servers to support multiple versions of YANG modules, and provide clients with some ability to select which versions of YANG modules they wish to interact with, subject to some reasonable constraints. This is the approach that this draft aims to address. It is worth noting that the idea of supporting multiple versions of an API is not new in the wider software industry, and there are many examples of where this approach has been successfully used.

### 4. Objectives

The goals of the schema version selection draft are:

- o To provide a mechanism where non-backwards-compatible changes and bug fixes can be made to YANG modules without forcing clients to immediately migrate to new versions of those modules as they get implemented.
- o To allow servers to support multiple versions of a particular YANG schema, and to allow clients to choose which YANG schema version to use when interoperating with the server. The aim here is to give operators more flexibility as to when they update their software.

- o To provide a mechanism to allow different YANG schema families (e.g., SDO models, OpenConfig models, Vendor models) to be supported by a server, and to allow clients to choose which YANG schema family is used to interoperate with the server.

The following points are non objective of this draft:

- o This draft does not provide a mechanism to allow clients to choose arbitrary sets of YANG module versions to interoperate with the server.
- o Servers are not required to concurrently support clients using different YANG schema families or versioned schema. A server MAY choose to only allow a single schema family or single versioned schema to be used by all clients.
- o There is no requirement for a server to support every published version of a YANG package, particularly if some package versions are backwards compatible. Clients are required to interoperate with backwards compatible updates of YANG modules. E.g., if a particular package was available in versions 1.0.0, 1.1.0, 1.2.0, 2.0.0, 3.0.0 and 3.1.0, then a server may choose to only support versions 1.2.0, 2.0.0, and 3.1.0, with the knowledge that all clients should be able to interoperate with the server.
- o There is no requirement to support all parts of all versioned schemas. For some nbc changes in modules, it is not possible for a server to support both the old and new module versions, and to convert between the two. Where appropriate deviations can be used, and otherwise an out of band mechanism is used to indicate where a mapping has failed.

## 5. Solution Overview

An overview the solution is as follows:

1. YANG packages are defined for the different versioned schema supported by a server:
  - \* Separate packages can be defined for different families of schema, e.g., SDO, OpenConfig, or vendor native.
  - \* Separate packages can be defined for each versioned schema within a schema family.
  - \* Separate packages may be defined for different datastores, if the datastores use different datastore schema. For example, a

different datastore schema, and hence package, might be used for <operational> vs the conventional datastores.

2. Each server advertises, via an operational data model:
  - \* All of the YANG packages that may be used during version selection. The packages can also be made available for offline consumption via instance data documents, as described in [I-D.rwilton-netmod-yang-packages].
  - \* Grouped sets of versioned schema, where each set defines the versioned schema used by each supported datastore, and each versioned schema is represented by a YANG package instance.
3. Each server supports configuration to:
  - \* Allow a client to configure which schema version set to use for the default NETCONF/RESTCONF connections.
  - \* Allow a client to configure additional separate NETCONF and RESTCONF protocol instances, which use different schema version sets on those protocol instances.
  - \* An RPC mechanism could also be defined to select schema, but is not currently discussed in this draft.
4. The server internally maps requests between the different protocol instances to the internal device implementation.

## 6. Version selection from a server perspective

The general premise of this solution is that servers generally implement one native schema, and the version selection scheme is used to support older version of that native schema and also foreign schema specified by external entities.

Overall the solution relies on the ability to map instance data between different schema versions. Depending on the scope of difference between the schema versions then some of these mappings may be very hard, or even impossible, to implement. Hence, there is still a strong incentive to try and minimize nbc changes between schema versions to minimize the mapping complexity.

Server implementations MUST serialize configuration requests across the different schema. The expectation is that this would be achieved by mapping all requests to the devices native schema version.

Datastore validation needs to be performed in two places, firstly in whichever schema a clients is interacting in, and secondly in the native schema for the device. This could have a negative performance impact.

Depending on the complexity of the mappings between schema versions, it may be necessary for the mappings to be stateful.

TODO - Figure out how hot fixes that slightly modify the schema are handled.

## 7. Version selection from a clients perspective

Clients can use configuration to choose which schema sets are available.

Clients cannot choose arbitrary individual YANG module versions, and are instead constrained by the versions that the server makes available.

Each client protocol connection is to one particular schema set. From that client session perspective it appears as if the client is interacting with a regular server. If the client queries YANG library that the version of YANG Library that is returned matches the schema set that is being used for that server instance.

The server may not support a schema with the exact version desired by the client, and they have to accept a later version that is backwards compatible with their desired version. Clients may also have to accept later schema versions that contain NBC fixes, although the assumption is that such nbc fixes should be designed to minimize the impact on clients.

There is no guarantee that servers will always be able to support all older schema versions. Deviations should be used where necessary to indicate that the server is unable to faithfully implement the older schema version.

If clients interact with a server using multiple versions, they should not expect that all data nodes in later module versions can always be backported to older schema versions. TODO - Specify how mapping errors can be reported to client.

## 8. Limitations of the solution

Not all schema conversions are possible. E.g. an impossible type conversion, or something has been removed. The solution is fundamentally limited by how the schemas actually change, this

solution does not provide a magic bullet that can solve all versioning issues.

## 9. Schema Version Selection YANG module

The YANG schema version selection YANG module is used by a device to report the schema-sets that are available, and to allow clients to choose which schema-set they wish to use.

Feature are used to allow servers to decide whether they allow the primary schema-set to be changed, and/or allow secondary schema-sets to be configured.

The primary schema-set is the datastore schema reported by YANG Library if a client connects to the device using the standard NETCONF/RESTCONF protocol numbers.

If secondary schema-sets are configured, then the client can choose whether NETCONF or RESTCONF is supported, which port numbers the protocols should run on (if available), and what RESTCONF root path prefix to use (e.g. if all of the RESTCONF protocol instances run on port 443).

Different schema-sets may support different datastores.

The "ietf-schema-version-selection" YANG module has the following structure:

```
module: ietf-schema-version-selection
  +--rw schema-selection
    +--rw schema-sets* [name]
      +--rw name string
      +--rw netconf! {secondary-schema-set}?
      | +--rw port? inet:port-number
      +--rw restconf! {secondary-schema-set}?
      | +--rw port? inet:port-number
      | +--rw root-path? inet:uri
      +--ro datastores* [datastore]
      | +--ro datastore ds:datastore-ref
      | +--ro package
      | | +--ro name?
      | | | -> /yanglib:yang-library/pkg:package/name
      | | +--ro version? leafref
      +--rw default-schema-set?
      | -> /schema-selection/schema-sets/name
      {default-schema-set}?
```

## 10. YANG Module

The YANG module definition for the module described in the previous sections.

```
<CODE BEGINS> file "ietf-schema-version-selection@2019-03-11.yang"
module ietf-schema-version-selection {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-schema-version-selection";
  prefix "ver-sel";

  import ietf-inet-types {
    prefix inet;
    reference "RFC 6991: Common YANG Data Types.";
  }
  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }
  import ietf-yang-library {
    prefix yanglib;
    reference "RFC 8525: YANG Library";
  }
  import ietf-yang-library-packages {
    prefix pkg;
    reference "draft-rwilton-netmod-yang-packages-01";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

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

    Author:   Reshad Rahman
              <mailto:rrahman@cisco.com>

    Author:   Rob Wilton
              <mailto:rwilton@cisco.com>";

  description
    "This module provide a data model to advertise and allow the
    selection of schema versions by clients."
```

Copyright (c) 2019 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

```
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
// RFC Ed.: replace XXXX with actual RFC number and remove this
// note.
revision 2019-03-11 {
  description
    "Initial revision";
  reference
    "RFC XXXX: YANG Schema Version Selection";
}

/*
 * Typedefs
 */

typedef yang-sem-ver {
  type string {
    pattern '\d+[\.]\d+[\.]\d+[mM]?';
  }
  description
    "Represents a YANG semantic version number.";
  reference
    "TODO - Should be defined by YANG versioning types module";
}

feature "default-schema-set" {
  description
    "Feature that allows clients to choose the default schema set
    to be used for clients that connect using the standard network
    configuration protocol port number or URL.

    Implementations may choose to only support this feature in
    <operational> to report the default-schema-set without
    allowing it to be configured.";
}
```

```
feature "secondary-schema-set" {
  description
    "Feature to choose if secondary schema sets may be configured
    by clients.

    Implementations may choose to only support this feature in
    <operational> to report secondary schema sets without
    allowing them to be configured.";
}

container schema-selection {
  description
    "YANG schema version selection";

  list schema-sets {
    key "name";

    description
      "All schema-sets that are available for client selection.";

    leaf name {
      type "string" {
        length "1..255";
      }
      description
        "The server assigned name of the schema-set.

        This should include the schema family, and appropriate
        versioning or release information";
    }
  }

  container netconf {
    if-feature "secondary-schema-set";

    presence "Make this schema-set available via NETCONF";
    description
      "NETCONF protocol settings for this schema set, if
      available";

    leaf port {
      type inet:port-number;
      description
        "The port numnber to use for interacting with this
        schema-set.  If not configured, then the port number is
        server allocated.";
      reference
        "RFC 6242: Using the NETCONF Protocol over SSH";
    }
  }
}
```



```
}

container restconf {
  if-feature "secondary-schema-set";

  presence
    "Make this schema-set available via RESTCONF";
  description
    "RESTCONF protocol settings for this schema set, if
    available";

  leaf port {
    type inet:port-number;
    default "443";
    description
      "The port numnber to use for interacting with this
      schema-set.  If not configured, then the port number
      defaults to the standard RESTCONF https port number of
      443";
    reference
      "RFC 8040: RESTCONF Protocol, section 2.1";
  }

  leaf root-path {
    type inet:uri;
    default "/restconf";
    description
      "The default root path to use to access the RESTCONF
      protocol instance for this schema-set";
  }
}

list datastores {
  key "datastore";
  config false;

  description
    "The list of datastores supported for this schema set";

  leaf datastore {
    type ds:datastore-ref;
    description
      "The datastore that this datastore schema is associated
      with";
    reference
      "RFC 8342: Network Management Datastore Architecture
      (NMDA)";
  }
}
```

```
    }

    container package {
      description
        "YANG package associated with this datastore schema";

      leaf name {
        type leafref {
          path "/yanglib:yang-library/pkg:package/pkg:name";
        }
        description
          "The name of the YANG package this schema relates to";
      }
      leaf version {
        type leafref {
          path '/yanglib:yang-library/'
            + 'pkg:package[pkg:name = current()/../name]/'
            + 'pkg:version';
        }

        description
          "The version of the YANG package this schema relates
            to";
      }
    }
  }
}

leaf default-schema-set {
  if-feature "default-schema-set";
  type leafref {
    path '/schema-selection/schema-sets/name';
  }
  description
    "Specifies the default schema-set used by this device. This
    is the set of datastore schema that is used if a client
    connects using the standard protocol port numbers and URLs";
}
}
}
<CODE ENDS>
```

## 11. Security Considerations

To be defined.

## 12. IANA Considerations

TODO - Add registrations for YANG modules defined in this draft.

## 13. Open Questions/Issues

All issues, along with the draft text, are currently being tracked at: TODO - URL

## 14. Acknowledgements

The ideas that formed this draft are based on discussions with the YANG versioning design team, and other members of the NETMOD WG.

## 15. References

### 15.1. Normative References

- [I-D.ietf-netconf-rfc7895bis]  
Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", draft-ietf-netconf-rfc7895bis-07 (work in progress), October 2018.
- [I-D.ietf-netmod-module-tags]  
Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", draft-ietf-netmod-module-tags-07 (work in progress), March 2019.
- [I-D.ietf-netmod-yang-instance-file-format]  
Lengyel, B. and B. Claise, "YANG Instance Data File Format", draft-ietf-netmod-yang-instance-file-format-02 (work in progress), February 2019.
- [I-D.rwilton-netmod-yang-packages]  
Wilton, R., "YANG Packages", draft-rwilton-netmod-yang-packages-00 (work in progress), December 2018.
- [I-D.verdt-netmod-yang-semver]  
Claise, B., Clarke, J., Rahman, R., Wilton, R., Lengyel, B., Sterne, J., and K. D'Souza, "YANG Semantic Versioning for Modules", draft-verdt-netmod-yang-semver-00 (work in progress), March 2019.
- [I-D.verdt-netmod-yang-versioning-reqs]  
Clarke, J., "YANG Module Versioning Requirements", draft-verdt-netmod-yang-versioning-reqs-02 (work in progress), November 2018.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [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>>.
- [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>>.
- [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>>.

## 15.2. Informative References

- [I-D.bierman-netmod-yang-package]  
Bierman, A., "The YANG Package Statement", draft-bierman-netmod-yang-package-00 (work in progress), July 2015.

- [I-D.ietf-netmod-artwork-folding]  
Watsen, K., Wu, Q., Farrel, A., and B. Claise, "Handling  
Long Lines in Inclusions in Internet-Drafts and RFCs",  
draft-ietf-netmod-artwork-folding-01 (work in progress),  
March 2019.
- [RFC8199] Bogdanovic, D., Claise, B., and C. Moberg, "YANG Module  
Classification", RFC 8199, DOI 10.17487/RFC8199, July  
2017, <<https://www.rfc-editor.org/info/rfc8199>>.

## Authors' Addresses

Robert Wilton  
Cisco Systems, Inc.

Email: [rwilton@cisco.com](mailto:rwilton@cisco.com)

Reshad Rahman  
Cisco Systems, Inc.

Email: [rrahman@cisco.com](mailto:rrahman@cisco.com)

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: September 12, 2019

T. Zhou  
G. Zheng  
Huawei  
E. Voit  
Cisco Systems  
A. Clemm  
Huawei  
A. Bierman  
YumaWorks  
March 11, 2019

Subscription to Multiple Stream Originators  
draft-zhou-netconf-multi-stream-originators-04

Abstract

This document describes the distributed data collection mechanism that allows multiple data streams to be managed using a single subscription. Specifically, multiple data streams are pushed directly to the collector without passing through a broker for internal consolidation.

Requirements Language

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

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 September 12, 2019.

## Copyright Notice

Copyright (c) 2019 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Use Cases . . . . .	3
2.1. Use Case 1: Data Collection from Devices with Main-board and Line-cards . . . . .	3
2.2. Use Case 2: IoT Data Collection . . . . .	4
3. Terminologies . . . . .	5
4. Solution Overview . . . . .	6
5. Subscription Decomposition . . . . .	8
6. Publication Composition . . . . .	9
7. Subscription State Change Notifications . . . . .	10
8. YANG Module . . . . .	10
9. IANA Considerations . . . . .	12
10. Security Considerations . . . . .	12
11. Acknowledgements . . . . .	13
12. References . . . . .	13
12.1. Normative References . . . . .	13
12.2. Informative References . . . . .	13
Appendix A. Change Log . . . . .	14
Authors' Addresses . . . . .	14

## 1. Introduction

Streaming telemetry refers to sending a continuous stream of operational data from a device to a remote receiver. This provides an ability to monitor a network from remote and to provide network analytics. Devices generate telemetry data and push that data to a collector for further analysis. By streaming the data, much better performance, finer-grained sampling, monitoring accuracy, and bandwidth utilization can be achieved than with polling-based alternatives.

YANG-Push [I-D.ietf-netconf-yang-push] defines a transport-independent subscription mechanism for datastore updates, in which a subscriber can subscribe to a stream of datastore updates from a server, or update provider. The current design involves subscription to a single push server. This conceptually centralized model encounters efficiency limitations in cases where the data sources are themselves distributed, such as line cards in a piece of network equipment. In such cases, it will be a lot more efficient to have each data source (e.g., each line card) originate its own stream of updates, rather than requiring updates to be tunneled through a central server where they are combined. What is needed is a distributed mechanism that allows to directly push multiple individual data substreams, without needing to first pass them through an additional processing stage for internal consolidation, but still allowing those substreams to be managed and controlled via a single subscription.

This document will describe such distributed data collection mechanism and how it can work by extending existing YANG-Push mechanism. The proposal is general enough to fit many scenarios.

## 2. Use Cases

### 2.1. Use Case 1: Data Collection from Devices with Main-board and Line-cards

For data collection from devices with main-board and line-cards, existing YANG-Push solutions consider only one push server typically reside in the main board. As shown in the following figure, data are collected from line cards and aggregate to the main board as one consolidated stream. So the main board can easily become the performance bottle-neck. The optimization is to apply the distributed data collection mechanism which can directly push data from line cards to a collector. On one hand, this will reduce the cost of scarce compute and memory resources on the main board for data processing and assembling. On the other hand, distributed data push can off-load the streaming traffic to multiple interfaces.



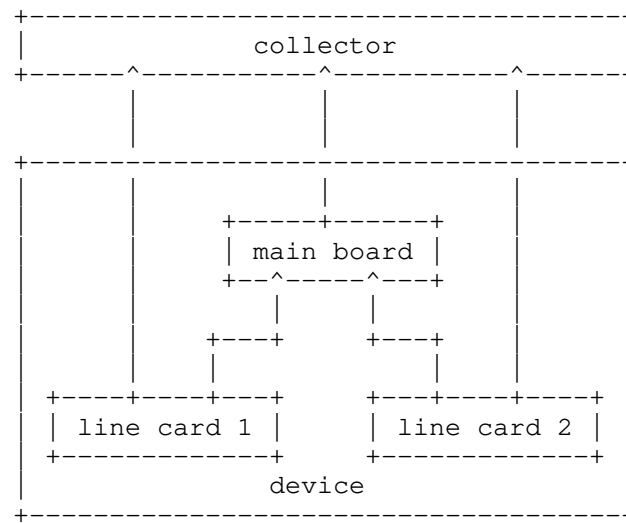


Fig. 1 Data Collection from Devices with Main-board and Line-cards

## 2.2. Use Case 2: IoT Data Collection

In the IoT data collection scenario, as shown in the following figure, collector usually cannot access to IoT nodes directly, but is isolated by the border router. So the collector subscribes data from the border router, and let the border router to disassemble the subscription to corresponding IoT nodes. The border router is typically the traffic convergence point. It's intuitive to treat the border router as a broker assembling the data collected from the IoT nodes and forwarding to the collector[I-D.ietf-core-coap-pubsub]. However, the border router is not so powerful on data assembling as a network device. It's more efficient for the collector, which may be a server or even a cluster, to assemble the subscribed data if possible. In this case, push servers that reside in IoT nodes can stream data to the collector directly while traffic only passes through the border router.

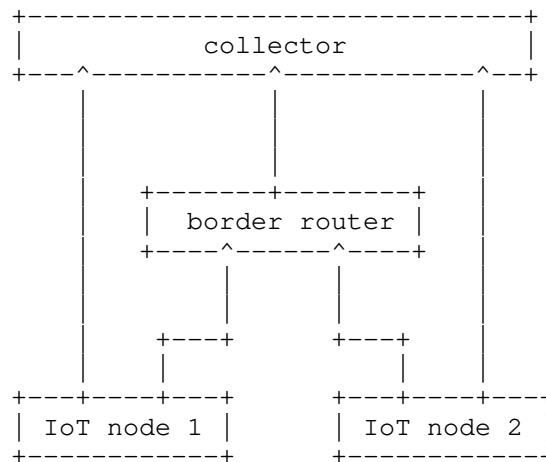


Fig. 2 IoT Data Collection

### 3. Terminologies

**Subscriber:** generates the subscription instructions to express what and how the collector want to receive the data

**Receiver:** is the target for the data publication.

**Publisher:** pushes data to the receiver according to the subscription information.

**Subscription Server:** which manages capabilities that it can provide to the subscriber.

**Global Subscription:** the subscription requested by the subscriber. It may be decomposed into multiple Component Subscriptions.

**Component Subscription:** is the subscription that defines the data from each individual telemetry source which is managed and controlled by a single Subscription Server.

**Global Capability:** is the overall subscription capability that the group of Publishers can expose to the Subscriber.

**Component Capability:** is the subscription capability that each Publisher can expose to the Subscriber.

**Master Publication Channel:** the session between the Master Publisher and the Receiver.

Agent Publication Channel: the session between the Agent Publisher and the Receiver.

#### 4. Solution Overview

All the use cases described in the previous section are very similar on the data subscription and publication mode, hence can be abstracted to the following generic distributed data collection framework, as shown in the following figure.

A Collector usually includes two components,

- o the Subscriber generates the subscription instructions to express what and how the collector want to receive the data;
- o the Receiver is the target for the data publication.

For one subscription, there may be one to many receivers. And the subscriber does not necessarily share the same address with the receivers.

In this framework, the Publisher pushes data to the receiver according to the subscription information. The Publisher has the Master role and the Agent role. Both the Master and the Agent include the Subscription Server which actually manages capabilities that it can provide to the subscriber.

The Master knows all the capabilities that the attached Agents and itself can provide, and exposes the Global Capability to the Collector. The Collector cannot see the Agents directly, so it will only send the Global Subscription information to the Master. The Master disassembles the Global Subscription to multiple Component Subscriptions, each involving data from a separate telemetry source. The Component Subscriptions are then distributed to the corresponding Agents.

When data streaming, the Publisher collects and encapsulates the packets per the Component Subscription, and pushes the piece of data which can serve directly to the designated data Collector. The Collector is able to assemble many pieces of data associated with one Global Subscription, and can also deduce the missing pieces of data.

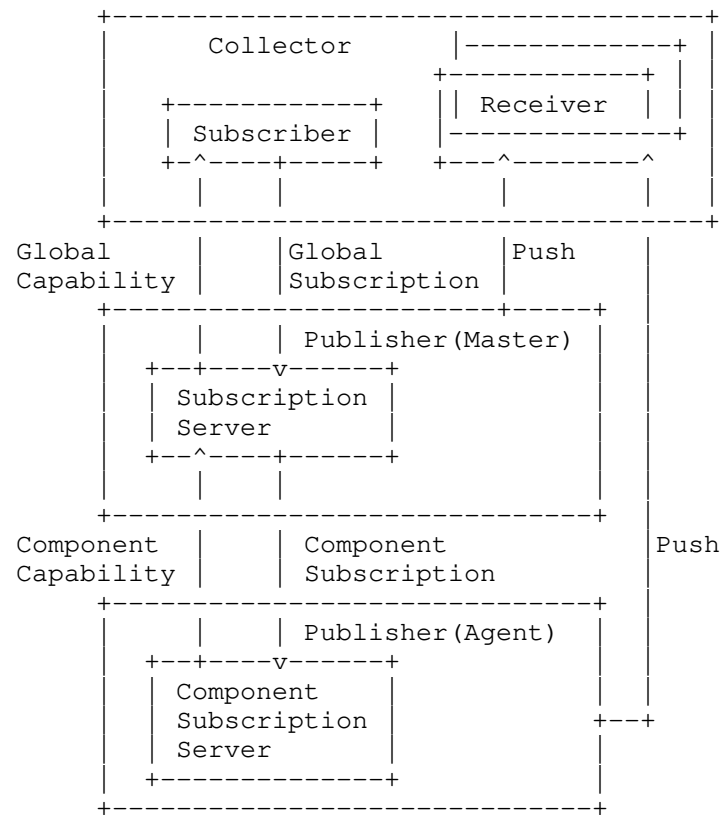


Fig. 3 The Generic Distributed Data Collection Framework

Master and Agents may interact with each other in several ways:

- o Agents need to have a registration or announcement handshake with the Master, so the Master is aware of them and of life-cycle events (such as Agent appearing and disappearing).
- o Contracts are needed between the Master and each Agent on the Component Capability, and the format for streaming data structure.
- o The Master relays the component subscriptions to the Agents.
- o The Agents indicate status of Component Subscriptions to the Master. The status of the overall subscription is maintained by the Master. The Master is also responsible for notifying the subscriber in case of any problems of Component Subscriptions.

Any technical mechanisms or protocols used for the coordination of operational information between Master and Agent is out-of-scope of the solution. We will need to instrument the results of this coordination on the Master Node.

## 5. Subscription Decomposition

Since Agents are invisible to the Collector, the Collector can only subscribe to the Master. This requires the Master to:

1. expose the Global Capability that can be served by multiple Publishers;
2. disassemble the Global Subscription to multiple Component Subscriptions, and distribute them to the corresponding telemetry sources;
3. notify on changes when portions of a subscription moving between different Agents over time.

To achieve the above requirements, the Master need a Global Capability description which is typically the YANG [RFC7950] data model. This global YANG model is provided as the contract between the Master and the Collector. Each Agent associating with the Master owns a local YANG model to describe the Component Capabilities which it can serve as part of the Global Capability. All the Agents need to know the namespace associated with the Master.

The Master also need a data structure, typically a Resource-Location Table, to keep track of the mapping between the resource and the corresponding location of the Subscription Server which commits to serve the data. When a Global Subscription request arrives, the Master will firstly extract the filter information from the request. Consequently, according to the Resource-Location Table, the Global Subscription can be disassembled into multiple Component Subscriptions, and the corresponding location can be associated.

The decision whether to decompose a Global Subscription into multiple Component Subscriptions rests with the Resource-Location Table. A Master can decide to not decompose a Global Subscription at all and push a single stream to the receiver, because the location information indicates the Global Subscription can be served locally by the Master. Similarly, it can decide to entirely decompose a Global Subscription into multiple Component Subscriptions that each push their own streams, but not from the Master. It can also decide to decompose the Global Subscription into several Component Subscriptions and retain some aspects of the Global Subscription itself, also pushing its own stream.

Component Subscriptions belonging to the same Global Subscription MUST NOT overlap. The combination of all Component Subscriptions MUST cover the same range of nodes as the Global Subscription. Also, the same subscription settings apply to each Component Subscription, i.e., the same receivers, the same time periods, the same encodings are applied to each Component Subscription per the settings of the Global Subscription.

Each Component Subscription in effect constitutes a full-fledged subscription, with the following constraints:

- o Component subscriptions are system-controlled, i.e. managed by the Master, not by the subscriber.
- o Component subscription settings such as time periods, dampening periods, encodings, receivers adopt the settings of their Global Subscription.
- o The life-cycle of the Component Subscription is tied to the life-cycle of the Global Subscription. Specifically, terminating/removing the Global Subscription results in termination/removal of Component Subscriptions.
- o The Component Subscriptions share the same Subscription ID as the Global Subscription.

## 6. Publication Composition

The Publisher collects data and encapsulates the packets per the Component Subscription. There are several potential encodings, including XML, JSON, CBOR and GPB. The format and structure of the data records are defined by the YANG schema, so that the composition at the Receiver can benefit from the structured and hierarchical data instance.

The Receiver is able to assemble many pieces of data associated with one subscription, and can also deduce the missing pieces of data. The Receiver recognizes data records associated with one subscription according the Subscription ID. Data records generated per one subscription are assigned with the same Subscription ID.

For the time series data stream, records are produced periodically from each stream originator. The message arrival time varies because of the distributed nature of the publication. The Receiver assembles data generated at the same time period based on the recording time consisted in each data record. In this case, time synchronization is required for all the Publishers.

To check the integrity of the data generated from different Publishers at the same time period, the Message Generator ID [I-D.ietf-netconf-notification-messages] is helpful. This requires the Subscriber to know the number of Component Subscriptions which the Global Subscription is decomposed to. For the dynamic subscription, the output of the "establish-subscription" and "modify-subscription" RPC defined in [I-D.ietf-netconf-subscribed-notifications] MUST include a list of Message Generator IDs to indicate how the Global Subscription is decomposed into several Component Subscriptions. The "subscription-started" and "subscription-modified" notification defined in [I-D.ietf-netconf-subscribed-notifications] MUST also include a list of Message Generator IDs to notify the current Publishers for the corresponding Global Subscription.

## 7. Subscription State Change Notifications

In addition to sending event records to receivers, the Master MUST also send subscription state change notifications [I-D.ietf-netconf-subscribed-notifications] when events related to subscription management have occurred. All the subscription state change notifications MUST be delivered by the Master Publication Channel which is the session between the Master Publisher and the Receiver.

When the subscription decomposition result changed, the "subscription-modified" notification will be sent to indicate the new a list of Publishers.

## 8. YANG Module

```
<CODE BEGINS> file "ietf-multiple-stream-originators@2019-03-11.yang"
module ietf-multiple-stream-originators {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-multiple-stream-originators";
  prefix mso;
  import ietf-subscribed-notifications {
    prefix sn;
  }

  organization "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>

    Editor:    Tianran Zhou
               <mailto:zhoutianran@huawei.com>
```

Editor: Guangying Zheng  
<mailto:zhengguangying@huawei.com>;

description

"Defines augmentation for ietf-subscribed-notifications to enable the distributed publication with single subscription.

Copyright (c) 2018 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

revision 2019-03-11 {

description

"Initial version";

reference

"RFC XXXX: Subscription to Multiple Stream Originators";

}

augment "/sn:subscription-started" {

description

"This augmentation allows MSO specific parameters to be exposed for a subscription.";

leaf message-generator-id {

type string;

description

"Software entity which created the message (e.g., linecard 1). This field is used to notify the collector the working originator";

}

}

augment "/sn:subscription-modified" {

description

"This augmentation allows MSO specific parameters to be exposed for a subscription.";

leaf message-generator-id {



```
    type string;
    description
      "Software entity which created the message (e.g., linecard 1).
      This field is used to notify the collector the working
      originator";
  }
}

augment "/sn:establish-subscription/sn:output" {
  description
    "This augmentation allows MSO specific parameters to be
    exposed for a subscription.";

  leaf message-generator-id {
    type string;
    description
      "Software entity which created the message (e.g., linecard 1).
      This field is used to notify the collector the working
      originator";
  }
}

augment "/sn:modify-subscription/sn:output" {
  description
    "This augmentation allows MSO specific parameters to be
    exposed for a subscription.";

  leaf message-generator-id {
    type string;
    description
      "Software entity which created the message (e.g., linecard 1).
      This field is used to notify the collector the working
      originator";
  }
}
}
<CODE ENDS>
```

## 9. IANA Considerations

TBD

## 10. Security Considerations

It's expected to reuse the existing secure transport layer protocols, such as TLS [RFC5246] and DTLS [RFC6347], to secure the telemetry stream.

## 11. Acknowledgements

TBD

## 12. References

### 12.1. Normative References

- [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>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [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>>.

### 12.2. Informative References

- [I-D.ietf-core-coap-pubsub] Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", draft-ietf-core-coap-pubsub-06 (work in progress), January 2019.
- [I-D.ietf-netconf-notification-messages] Voit, E., Birkholz, H., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-05 (work in progress), February 2019.
- [I-D.ietf-netconf-subscribed-notifications] Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Event Notifications", draft-ietf-netconf-subscribed-notifications-23 (work in progress), February 2019.

[I-D.ietf-netconf-yang-push]

Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-22 (work in progress), February 2019.

#### Appendix A. Change Log

(To be removed by RFC editor prior to publication)

v01

- o Minor revision on Subscription Decomposition
- o Revised terminologies
- o Removed most implementation related text
- o Place holder of two sections: Subscription Management, and Notifications on Subscription State Changes

v02

- o Revised section 4 and 5. Moved them from appendix to the main text.

v03

- o Added a section for Terminologies.
- o Added a section for Subscription State Change Notifications.
- o Improved the Publication Composition section by adding a method to check the integrity of the data generated from different Publishers at the same time period.
- o Revised the solution overview for a more clear description.

v04

- o Added the YANG data model for the proposed augment.

Authors' Addresses

Tianran Zhou  
Huawei  
156 Beiqing Rd., Haidian District  
Beijing  
China

Email: zhoutianran@huawei.com

Guangying Zheng  
Huawei  
101 Yu-Hua-Tai Software Road  
Nanjing, Jiangsu  
China

Email: zhengguangying@huawei.com

Eric Voit  
Cisco Systems  
United States of America

Email: evoit@cisco.com

Alexander Clemm  
Huawei  
2330 Central Expressway  
Santa Clara, California  
United States of America

Email: alexander.clemm@huawei.com

Andy Bierman  
YumaWorks  
United States of America

Email: andy@yumaworks.com