Common YANG Data Types for Cryptography
draft-ietf-netconf-crypto-types-10

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:

- "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:

- "2019-07-02" --> the publication date of this draft

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

- 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."
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 January 3, 2020.

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 .................................... 5
3. Security Considerations ............................... 48
   3.1. Support for Algorithms .......................... 48
   3.2. No Support for CRMF .............................. 48
   3.3. Access to Data Nodes ............................ 48
4. IANA Considerations .................................... 50
   4.1. The IETF XML Registry ........................... 50
   4.2. The YANG Module Names Registry .................. 50
5. References ............................................. 50
   5.1. Normative References ............................. 50
   5.2. Informative References ........................... 53
Appendix A. Examples ...................................... 56
   A.1. The "asymmetric-key-pair-with-certs-grouping" Grouping . 56
   A.2. The "generate-certificate-signing-request" Action ........ 58
   A.3. The "certificate-expiration" Notification ............ 59
Appendix B. Change Log .................................... 60
   B.1. I-D to 00 ....................................... 60
   B.2. 00 to 01 ........................................ 60
   B.3. 01 to 02 ........................................ 60
   B.4. 02 to 03 ........................................ 61
   B.5. 03 to 04 ........................................ 61
   B.6. 04 to 05 ........................................ 62
   B.7. 05 to 06 ........................................ 62
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.

```.yang
module: ietf-crypto-types

  grouping symmetric-key-grouping
    +-- algorithm encryption-algorithm-t
    +-- (key-type)
      +--:(key)
      |   +-- key? binary
      +--:(hidden-key)
      +-- hidden-key? empty

  grouping public-key-grouping
    +-- algorithm asymmetric-key-algorithm-t
    +-- public-key binary

  grouping asymmetric-key-pair-grouping
    +-- algorithm asymmetric-key-algorithm-t
    +-- public-key binary
    +-- (private-key-type)
      +--:(private-key)
      |   +-- private-key? binary
      +--:(hidden-private-key)
      +-- hidden-private-key? empty

  grouping trust-anchor-cert-grouping
    +-- cert? trust-anchor-cert-cms
```

Watsen & Wang            Expires January 3, 2020                [Page 3]
+---n certificate-expiration
  +-- expiration-date    yang:date-and-time
grouping trust-anchor-certs-grouping
  +-- cert*                     trust-anchor-cert-cms
+---n certificate-expiration
  +-- expiration-date    yang:date-and-time
grouping end-entity-cert-grouping
  +-- cert*                     end-entity-cert-cms
+---n certificate-expiration
  +-- expiration-date    yang:date-and-time
grouping end-entity-certs-grouping
  +-- cert?                     end-entity-cert-cms
+---n certificate-expiration
  +-- expiration-date    yang:date-and-time
grouping asymmetric-key-pair-with-cert-grouping
  +-- algorithm
    |       asymmetric-key-algorithm-t
    +-- public-key                              binary
    +-- (private-key-type)
      |       (private-key)
      |       |       private-key?                      binary
      |       +--:(hidden-private-key)
      |       |       hidden-private-key?               empty
      +-- cert?                                   end-entity-cert-cms
+---n certificate-expiration
  +-- expiration-date    yang:date-and-time
+-- generate-certificate-signing-request
  +-- w input
    |       w subject       binary
    |       w attributes?   binary
  +-- ro output
    +-- ro certificate-signing-request    binary
grouping asymmetric-key-pair-with-certs-grouping
  +-- algorithm
    |       asymmetric-key-algorithm-t
    +-- public-key                              binary
    +-- (private-key-type)
      |       (private-key)
      |       |       private-key?                      binary
      |       +--:(hidden-private-key)
      |       |       hidden-private-key?               empty
      +-- certificates
        +-- certificate* [name]
          |       name?                     string
          |       cert?                     end-entity-cert-cms
          +--n certificate-expiration
            +-- expiration-date    yang:date-and-time
            +-- generate-certificate-signing-request
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@2019-07-02.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.

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 (https://trustee.ietf.org/license-info).

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

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’, ‘SHALL NOT’, ‘SHOULD’, ‘SHOULD NOT’, ‘RECOMMENDED’, ‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as described in BCP 14 (RFC 2119) when, and only when, they appear in all capitals, as shown here."

revision 2019-07-02 {
  description
    "Initial version";
  reference
    "RFC XXXX: Common YANG Data Types for Cryptography";
}

().'/**********************
/* Identities for Hash Algorithms */
/*********************/

typedef hash-algorithm-t {
  type union {
    type uint16;
    type enumeration {
      enum NONE {
        value 0;
        description
          "Hash algorithm is NULL.";
      } enum shal {
        value 1;
        status obsolete;
        description
          "The SHA1 algorithm.";
      } enum sha2 {
        value 2;
        description
          "The SHA2 algorithm.";
      } enum sha3 {
        value 3;
        description
enum sha-224 {
  value 2;
  description
  "The SHA-224 algorithm.";
  reference
  "RFC 6234: US Secure Hash Algorithms.";
}

enum sha-256 {
  value 3;
  description
  "The SHA-256 algorithm.";
  reference
  "RFC 6234: US Secure Hash Algorithms.";
}

enum sha-384 {
  value 4;
  description
  "The SHA-384 algorithm.";
  reference
  "RFC 6234: US Secure Hash Algorithms.";
}

enum sha-512 {
  value 5;
  description
  "The SHA-512 algorithm.";
  reference
  "RFC 6234: US Secure Hash Algorithms.";
}

enum shake-128 {
  value 6;
  description
  "The SHA3 algorithm with 128-bits output.";
  reference
  "National Institute of Standards and Technology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015.";
}

enum shake-224 {
  value 7;
  description
  "The SHA3 algorithm with 224-bits output.";
  reference
  "National Institute of Standards and Technology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015.";
}
enum shake-256 {
  value 8;
  description
    "The SHA3 algorithm with 256-bits output.";
  reference
    "National Institute of Standards and Technology,
    SHA-3 Standard: Permutation-Based Hash and
    Extendable-Output Functions, FIPS PUB 202, DOI
    10.6028/NIST.FIPS.202, August 2015.";
}
enum shake-384 {
  value 9;
  description
    "The SHA3 algorithm with 384-bits output.";
  reference
    "National Institute of Standards and Technology,
    SHA-3 Standard: Permutation-Based Hash and
    Extendable-Output Functions, FIPS PUB 202, DOI
    10.6028/NIST.FIPS.202, August 2015.";
}
enum shake-512 {
  value 10;
  description
    "The SHA3 algorithm with 512-bits output.";
  reference
    "National Institute of Standards and Technology,
    SHA-3 Standard: Permutation-Based Hash and
    Extendable-Output Functions, FIPS PUB 202, DOI
    10.6028/NIST.FIPS.202, August 2015.";
}
default "0";
description
  "The uint16 filed shall be set by individual protocol families
  according to the hash algorithm value assigned by IANA. The
  setting is optional and by default is 0. The enumeration
  filed is set to the selected hash algorithm.";
}
type union {
    type uint16;
    type enumeration {
        enum NONE {
            value 0;
            description
                "Asymetric key algorithm is NULL.";
        }
        enum rsa1024 {
            value 1;
            description
                "The RSA algorithm using a 1024-bit key.";
            reference
                "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
        }
        enum rsa2048 {
            value 2;
            description
                "The RSA algorithm using a 2048-bit key.";
            reference
                "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
        }
        enum rsa3072 {
            value 3;
            description
                "The RSA algorithm using a 3072-bit key.";
            reference
                "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
        }
        enum rsa4096 {
            value 4;
            description
                "The RSA algorithm using a 4096-bit key.";
            reference
                "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
        }
        enum rsa7680 {
            value 5;
            description
                "The RSA algorithm using a 7680-bit key.";
            reference
                "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2.";
        }
    }
}
enum rsa15360 {
    value 6;
    description
    "The RSA algorithm using a 15360-bit key.";
    reference
    "RFC 8017:
    PKCS #1: RSA Cryptography Specifications Version 2.2.";
}
enum secp192r1 {
    value 7;
    description
    "The asymmetric algorithm using a NIST P192 Curve.";
    reference
    "RFC 6090:
    Fundamental Elliptic Curve Cryptography Algorithms.
    RFC 5480:
    Elliptic Curve Cryptography Subject Public Key
    Information.";
}
enum secp224r1 {
    value 8;
    description
    "The asymmetric algorithm using a NIST P224 Curve.";
    reference
    "RFC 6090:
    Fundamental Elliptic Curve Cryptography Algorithms.
    RFC 5480:
    Elliptic Curve Cryptography Subject Public Key
    Information.";
}
enum secp256r1 {
    value 9;
    description
    "The asymmetric algorithm using a NIST P256 Curve.";
    reference
    "RFC 6090:
    Fundamental Elliptic Curve Cryptography Algorithms.
    RFC 5480:
    Elliptic Curve Cryptography Subject Public Key
    Information.";
}
enum secp384r1 {
    value 10;
    description
    "The asymmetric algorithm using a NIST P384 Curve.";
    reference
    "RFC 6090:
    Fundamental Elliptic Curve Cryptography Algorithms.
    RFC 5480:
    Elliptic Curve Cryptography Subject Public Key
    Information.";
}
enum secp521r1 {
  value 11;
  description "The asymmetric algorithm using a NIST P521 Curve."
  reference "RFC 5480: Elliptic Curve Cryptography Subject Public Key Information.";
}

enum x25519 {
  value 12;
  description "The asymmetric algorithm using a x.25519 Curve."
  reference "RFC 7748: Elliptic Curves for Security.";
}

enum x448 {
  value 13;
  description "The asymmetric algorithm using a x.448 Curve."
  reference "RFC 7748: Elliptic Curves for Security.";
}

default "0";

description "The uint16 filed shall be set by individual protocol families according to the asymmetric key algorithm value assigned by IANA. The setting is optional and by default is 0. The enumeration filed is set to the selected asymmetric key algorithm.";

typedef mac-algorithm-t {
  type union {
/* Identities for MAC Algorithms */
/*****************************/
type uint16;
type enumeration {
enum NONE {
    value 0;
    description "mac algorithm is NULL.";
}
enum hmac-sha1 {
    value 1;
    description "Generating MAC using SHA1 hash function";
    reference "RFC 3174: US Secure Hash Algorithm 1 (SHA1)";
}
enum hmac-sha1-96 {
    value 2;
    description "Generating MAC using SHA1 hash function";
    reference "RFC 2404: The Use of HMAC-SHA-1-96 within ESP and AH";
}
enum hmac-sha2-224 {
    value 3;
    description "Generating MAC using SHA2 hash function";
    reference "RFC 6234: US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)";
}
enum hmac-sha2-256 {
    value 4;
    description "Generating MAC using SHA2 hash function";
    reference "RFC 6234: US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)";
}
enum hmac-sha2-256-128 {
    value 5;
    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";
}
enum hmac-sha2-384 {
    value 6;
description
"Generating a 384 bits MAC using SHA2 hash function";
reference
"RFC 6234: US Secure Hash Algorithms
(SHA and SHA-based HMAC and HKDF)";
}
enum hmac-sha2-384-192 {
  value 7;
  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";
}
enum hmac-sha2-512 {
  value 8;
  description
  "Generating a 512 bits MAC using SHA2 hash function";
  reference
  "RFC 6234: US Secure Hash Algorithms
  (SHA and SHA-based HMAC and HKDF)";
}
enum hmac-sha2-512-256 {
  value 9;
  description
  "Generating a 512 bits MAC using SHA2 hash function and
  truncate it to 256 bits";
  reference
  "RFC 4868: Using HMAC-SHA-256, HMAC-SHA-384,
  and HMAC-SHA-512 with IPsec";
}
enum aes-128-gmac {
  value 10;
  description
  "Generating 128-bit 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";
}
enum aes-192-gmac {
  value 11;
  description
  "Generating 192-bit 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";

} enum aes-256-gmac {
  value 12;
  description
  "Generating 256-bit 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";
}

} enum aes-cmac-96 {
  value 13;
  description
  "Generating 96-bit MAC using Advanced Encryption Standard (AES) Cipher-based Message Authentication Code (CMAC)";
  reference
  "RFC 4494: The AES-CMAC Algorithm and its Use with IPsec";
}

} enum aes-cmac-128 {
  value 14;
  description
  reference
  "RFC 4494: The AES-CMAC Algorithm and its Use with IPsec";
}

} enum sha1-des3-kd {
  value 15;
  description
  "Generating MAC using triple DES encryption function";
  reference
  "RFC 3961: Encryption and Checksum Specifications for Kerberos 5";
}
typedef encryption-algorithm-t {
    type union {
        type uint16;
        type enumeration {
            enum NONE {
                value 0;
                description
                "Encryption algorithm is NULL.";
            }
            enum aes-128-cbc {
                value 1;
                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)";
            }
            enum aes-192-cbc {
                value 2;
                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)";
            }
            enum aes-256-cbc {
                value 3;
                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)";
            }
        }
    }
}
a key length of 256 bits;
    reference
    "RFC 3565: Use of the Advanced Encryption Standard (AES)
    Encryption Algorithm in Cryptographic Message Syntax
    (CMS)";
}

enum aes-128-ctr {
  value 4;
  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)";
}

enum aes-192-ctr {
  value 5;
  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)";
}

enum aes-256-ctr {
  value 6;
  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)";
}

enum des3-cbc-sha1-kd {
  value 7;
  description
  "Encrypt message with 3DES algorithm in CBC mode
  with sha1 function for key derivation";
  reference
  "RFC 3961:
  Encryption and Checksum Specifications for
  Kerberos 5";
enum rc4-hmac {
    value 8;
    description
    "Encrypt message with rc4 algorithm";
    reference
    "RFC 4757:
    The RC4-HMAC Kerberos Encryption Types Used by
    Microsoft Windows";
}

enum rc4-hmac-exp {
    value 9;
    description
    "Encrypt message with rc4 algorithm that is exportable";
    reference
    "RFC 4757:
    The RC4-HMAC Kerberos Encryption Types Used by
    Microsoft Windows";
}

default "0";

description
"The uint16 filed shall be set by individual protocol
families according to the encryption algorithm value
assigned by IANA. The setting is optional and by default
is 0. The enumeration filed is set to the selected
encryption algorithm.";

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

typedef encryption-and-mac-algorithm-t {
    type union {
        type uint16;
        type enumeration {
            enum NONE {
                value 0;
                description
                "Encryption and MAC algorithm is NULL.";
                reference
                "None";
            }
            enum aes-128-ccm {
                value 1;
                description
            }
        }
    }
}

Watsen & Wang Expires January 3, 2020 [Page 17]
"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)";
}

enum aes-192-ccm {
  value 2;
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)";
}

enum aes-256-ccm {
  value 3;
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)";
}

enum aes-128-gcm {
  value 4;
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)";
}

enum aes-192-gcm {
  value 5;
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)";
}
enum aes-256-gcm {
  value 6;
  description
  "Encrypt message with AES algorithm in GCM
  mode with a key length of 256 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)";
}

enum chacha20-poly1305 {
  value 7;
  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";
}

default "0";

typedef signature-algorithm-t {
  type union {
    type uint16;
    type enumeration {
      enum NONE {
        value 0;
        description
        "Signature algorithm is NULL";
      }
      enum dsa-sha1 {
        value 1;
        description
        "Signature algorithm is DSA SHA1";
      }
    }
  }
}
"The signature algorithm using DSA algorithm with SHA1 hash algorithm";
reference
"RFC 4253:
The Secure Shell (SSH) Transport Layer Protocol";
}
enum rsassa-pkcs1-sha1 {
  value 2;
  description
  "The signature algorithm using RSASSA-PKCS1-v1_5 with the SHA1 hash algorithm.";
  reference
  "RFC 4253:
The Secure Shell (SSH) Transport Layer Protocol";
}
enum rsassa-pkcs1-sha256 {
  value 3;
  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";
}
enum rsassa-pkcs1-sha384 {
  value 4;
  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";
}
enum rsassa-pkcs1-sha512 {
  value 5;
  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:
enum rsassa-pss-rsae-sha256 {
  value 6;
  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";
}

enum rsassa-pss-rsae-sha384 {
  value 7;
  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";
}

enum rsassa-pss-rsae-sha512 {
  value 8;
  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";
}

enum rsassa-pss-pss-sha256 {
  value 9;
  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:
enum rsassa-pss-pss-sha384 {
    value 10;
    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";
}

enum rsassa-pss-pss-sha512 {
    value 11;
    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";
}

enum ecdsa-secp256r1-sha256 {
    value 12;
    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";
}

enum ecdsa-secp384r1-sha384 {
    value 13;
    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";
}
enum ecdsa-secp521r1-sha512 {
  value 14;
  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"
}

enum ed25519 {
  value 15;
  description
  "The signature algorithm using EdDSA with curve x25519";
  reference
  "RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)";
}

enum ed25519-cts {
  value 16;
  description
  "The signature algorithm using EdDSA with curve x25519 with phflag = 0";
  reference
  "RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)";
}

enum ed25519-ph {
  value 17;
  description
  "The signature algorithm using EdDSA with curve x25519 with phflag = 1";
  reference
  "RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)";
}

enum ed25519-sha512 {
  value 18;
  description
  "The signature algorithm using EdDSA with curve x25519 with SHA512 hash algorithm";
  reference
  "RFC 5656: Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer"
}
and SHA-512 function";
reference
"RFC 8419:
Use of Edwards-Curve Digital Signature Algorithm
(EdDSA) Signatures in the Cryptographic Message
Syntax (CMS)"
)
enum ed448 {
  value 19;
description
"The signature algorithm using EdDSA with curve x448";
reference
"RFC 8032:
Edwards-Curve Digital Signature Algorithm (EdDSA)"
}
enum ed448-ph {
  value 20;
description
"The signature algorithm using EdDSA with curve x448
  and with PH being SHAKE256(x, 64) and phflag being 1"
reference
"RFC 8032:
Edwards-Curve Digital Signature Algorithm (EdDSA)"
}
enum ed448-shake256 {
  value 21;
description
"The signature algorithm using EdDSA with curve x448
  and SHAKE-256 function";
reference
"RFC 8419:
Use of Edwards-Curve Digital Signature Algorithm
(EdDSA) Signatures in the Cryptographic Message
Syntax (CMS)"
}
enum ed448-shake256-len {
  value 22;
description
"The signature algorithm using EdDSA with curve x448
  and SHAKE-256 function and a customized hash output"
reference
"RFC 8419:
Use of Edwards-Curve Digital Signature Algorithm
(EdDSA) Signatures in the Cryptographic Message
Syntax (CMS)"
}
enum rsa-sha2-256 {
  value 23;
description
"The signature algorithm using RSA with SHA2 function for SSH protocol";
reference
"RFC 8332: Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol";
}
enum rsa-sha2-512 {
  value 24;
  description
  "The signature algorithm using RSA with SHA2 function for SSH protocol";
  reference
  "RFC 8332: Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol";
}
enum eccsi {
  value 25;
  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)";
}
}

default "0";

description
"The uint16 filed shall be set by individual protocol families according to the signature algorithm value assigned by IANA. The setting is optional and by default is 0. The enumeration filed is set to the selected signature algorithm.";

/**************************************************************************************/
/* Identities for key exchange algorithms */
/**************************************************************************************/
typedef key-exchange-algorithm-t {
  type union {
    type uint16;
    type enumeration {
      enum NONE {

value 0;
  description
  "Key exchange algorithm is NULL.";
}
enum psk-only {
  value 1;
  description
  "Using Pre-shared key for authentication and key exchange";
  reference
  "RFC 4279: Pre-Shared Key cipher suites for Transport Layer Security (TLS)";
}
enum dhe-ffdhe2048 {
  value 2;
  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)";
}
enum dhe-ffdhe3072 {
  value 3;
  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)";
}
enum dhe-ffdhe4096 {
  value 4;
  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)";
}
enum dhe-ffdhe6144 {
  value 5;
  description
  "Ephemeral Diffie Hellman key exchange with 6144 bit
enum dhe-ffdhe8192 {
    value 6;
    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)";
}

enum psk-dhe-ffdhe2048 {
    value 7;
    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";
}

enum psk-dhe-ffdhe3072 {
    value 8;
    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";
}

enum psk-dhe-ffdhe4096 {
    value 9;
    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";
enum psk-dhe-ffdhe6144 {
  value 10;
  description  "Key exchange using pre-shared key with Diffie-Hellman key generation mechanism, where the DH group is FFDHE6144";
}

enum psk-dhe-ffdhe8192 {
  value 11;
  description  "Key exchange using pre-shared key with Diffie-Hellman key generation mechanism, where the DH group is FFDHE8192";
}

enum ecdhe-secp256r1 {
  value 12;
  description  "Ephemeral Diffie Hellman key exchange with elliptic group over curve secp256r1";
}

enum ecdhe-secp384r1 {
  value 13;
  description  "Ephemeral Diffie Hellman key exchange with elliptic group over curve secp384r1";
}

enum ecdhe-secp521r1 {
  value 14;
  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";
}
enum ecdhe-x25519 {
  value 15;
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";
}
enum ecdhe-x448 {
  value 16;
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";
}
enum psk-ecdhe-secp256r1 {
  value 17;
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";
}
enum psk-ecdhe-secp384r1 {
  value 18;
description
  "Key exchange using pre-shared key with elliptic
group-based Ephemeral Diffie Hellman key exchange
over curve secp384r1";
reference
enum psk-ecdhe-secp521r1 {
    value 19;
    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";
}

enum psk-ecdhe-x25519 {
    value 20;
    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";
}

enum psk-ecdhe-x448 {
    value 21;
    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";
}

enum diffie-hellman-group14-sha1 {
    value 22;
    description "Using DH group14 and SHA1 for key exchange";
    reference "RFC 4253:
The Secure Shell (SSH) Transport Layer Protocol";
}

enum diffie-hellman-group14-sha256 {
    value 23;
    description
enum diffie-hellman-group15-sha512 {
    value 24;
    description "Using DH group15 and SHA-512 for key exchange";
    reference "RFC 8268:
        More Modular Exponentiation (MODP) Diffie-Hellman (DH)
        Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

enum diffie-hellman-group16-sha512 {
    value 25;
    description "Using DH group16 and SHA-512 for key exchange";
    reference "RFC 8268:
        More Modular Exponentiation (MODP) Diffie-Hellman (DH)
        Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

enum diffie-hellman-group17-sha512 {
    value 26;
    description "Using DH group17 and SHA-512 for key exchange";
    reference "RFC 8268:
        More Modular Exponentiation (MODP) Diffie-Hellman (DH)
        Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

enum diffie-hellman-group18-sha512 {
    value 27;
    description "Using DH group18 and SHA-512 for key exchange";
    reference "RFC 8268:
        More Modular Exponentiation (MODP) Diffie-Hellman (DH)
        Key Exchange (KEX) Groups for Secure Shell (SSH)";
}

enum ecdh-sha2-secp256r1 {
    value 28;
    description "Elliptic curve-based Diffie Hellman key exchange over
curve ecp256r1 and using SHA2 for MAC generation";
    reference
enum ecdh-sha2-secp384r1 {
    value 29;
    description
    "Elliptic curve-based Diffie Hellman key exchange over
    curve ecp384r1 and using SHA2 for MAC generation";
    reference
    "RFC 6239:
    Suite B Cryptographic Suites for Secure Shell (SSH)";
}
enum ecdh-x25519-x9.63-sha256 {
    value 30;
    description
    "Elliptic curve-based Diffie Hellman key exchange over
    curve x.25519 and using ANSI x9.63 with SHA256 as KDF";
    reference
    "RFC 8418:
    Use of the Elliptic Curve Diffie-Hellman Key Agreement
    Algorithm with X25519 and X448 in the Cryptographic
    Message Syntax (CMS)";
}
enum ecdh-x25519-x9.63-sha384 {
    value 31;
    description
    "Elliptic curve-based Diffie Hellman key exchange over
    curve x.25519 and using ANSI x9.63 with SHA384 as KDF";
    reference
    "RFC 8418:
    Use of the Elliptic Curve Diffie-Hellman Key Agreement
    Algorithm with X25519 and X448 in the Cryptographic
    Message Syntax (CMS)";
}
enum ecdh-x25519-x9.63-sha512 {
    value 32;
    description
    "Elliptic curve-based Diffie Hellman key exchange over
    curve x.25519 and using ANSI x9.63 with SHA512 as KDF";
    reference
    "RFC 8418:
    Use of the Elliptic Curve Diffie-Hellman Key Agreement
    Algorithm with X25519 and X448 in the Cryptographic
    Message Syntax (CMS)";
}
enum ecdh-x25519-hkdf-sha256 {
    value 33;
    description
"Elliptic curve-based Diffie Hellman key exchange over curve x.25519 and using HKDF with SHA256 as KDF";
reference
"RFC 8418:
Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS);"
}
enum ecdh-x25519-hkdf-sha384 {
  value 34;
description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.25519 and using HKDF with SHA384 as KDF";
reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS);"
}
enum ecdh-x25519-hkdf-sha512 {
  value 35;
description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.25519 and using HKDF with SHA512 as KDF";
reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS);"
}
enum ecdh-x448-x9.63-sha256 {
  value 36;
description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.448 and using ANSI x9.63 with SHA256 as KDF";
reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS);"
}
enum ecdh-x448-x9.63-sha384 {
  value 37;
description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.448 and using ANSI x9.63 with SHA384 as KDF";
reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS);"
}
Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS);

enum ecdh-x448-x9.63-sha512 {
  value 38;
  description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.448 and using ANSI x9.63 with SHA512 as KDF";
  reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)";
}

enum ecdh-x448-hkdf-sha256 {
  value 39;
  description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.448 and using HKDF with SHA256 as KDF";
  reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)";
}

enum ecdh-x448-hkdf-sha384 {
  value 40;
  description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.448 and using HKDF with SHA384 as KDF";
  reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)";
}

enum ecdh-x448-hkdf-sha512 {
  value 41;
  description
  "Elliptic curve-based Diffie Hellman key exchange over curve x.448 and using HKDF with SHA512 as KDF";
  reference
  "RFC 8418:
  Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)";
}
enum rsaes-oaep {
  value 42;
  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.";
}
enum rsaes-pkcs1-v1_5 {
  value 43;
  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.";
}
}
default "0";
description
  "The uint16 filed shall be set by individual protocol
families according to the key exchange algorithm value
assigned by IANA. The setting is optional and by default
is 0. The enumeration filed is set to the selected key
exchange algorithm."
}
/*****************************/
/* 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),
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.";
}
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";
}

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."
}

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

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 symmetric-key-grouping {
    description
        "A symmetric key and algorithm.";
    leaf algorithm {
        type encryption-algorithm-t;
        mandatory true;
        description
            "The algorithm to be used when generating the key.";
        reference
            "RFC CCCC: Common YANG Data Types for Cryptography";
    }
    choice key-type {
        mandatory true;
        description
            "Choice between key types.";
        leaf key {
            nacm:default-deny-all;
            type binary;
            description
                "The binary value of the key. The interpretation of
                the value is defined by ‘algorithm’. For example,
                FIXME.";
            reference
                "RFC XXXX: FIXME";
        }
        leaf hidden-key {
            nacm:default-deny-write;
            type empty;
            description
                "A permanently hidden key. How such keys are created
                is outside the scope of this module.";
        }
    }
}

grouping public-key-grouping {
    description
        "A public key and its associated algorithm.";
    leaf algorithm {

grouping asymmetric-key-pair-grouping {
    description "A private key and its associated public key and algorithm.";
    uses public-key-grouping;
    choice private-key-type {
        mandatory true;
        description "Choice between key types.";
        leaf private-key {
            nacm:default-deny-all;
            type binary;
            description "The value of the binary key. The key’s value is interpreted by the 'algorithm'. For example, a DSA key is an integer, an RSA key is represented as RSAPrivateKey as defined in RFC 8017, and an ECC key is represented as ECPrivateKey as defined in RFC 5915.";
        }
        leaf hidden-private-key {
            Watsen & Wang Expires January 3, 2020 [Page 41]
nacm:default-deny-write;
type empty;
description
"A permanently hidden key. How such keys are created
is outside the scope of this module."
}
}
}
grouping trust-anchor-cert-grouping {

description
"A trust anchor certificate, and a notification for when
it is about to (or already has) expire."
leaf cert {
    nacm:default-deny-write;
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 trust-anchor-certs-grouping {

description
"A list of trust anchor certificates, and a notification
for when one is about to (or already has) expire."
leaf-list cert {
    nacm:default-deny-write;
type trust-anchor-cert-cms;
description
"The binary certificate data for this certificate.";
reference
"RFC YYYYY: 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
  "An end entity certificate, and a notification for when it is about to (or already has) expire. Implementations SHOULD assert that, where used, the end entity certificate contains the expected public key.";
  leaf cert {
    nacm:default-deny-write;
    type end-entity-cert-cms;
    description
    "The binary certificate data for this certificate.";
    reference
    "RFC YYYYY: 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-certs-grouping {
    description
    "A list of end entity certificates, and a notification for
    when one is about to (or already has) expire.";
    leaf-list cert {
        nacm:default-deny-write;
        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-cert-grouping {
    description
    "A private/public key pair and an associated certificate.
    Implementations SHOULD assert that certificates contain
    the matching public key.";
    uses asymmetric-key-pair-grouping;
    uses end-entity-cert-grouping;
    action generate-certificate-signing-request {
        nacm:default-deny-all;
        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; // FIXME: does this need to be mandatory?
    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:"
  }
}
grouping asymmetric-key-pair-with-certs-grouping {
  description
  "A private/public key pair and associated certificates. Implementations SHOULD assert that certificates contain the matching public key.";
  uses asymmetric-key-pair-grouping;
  container certificates {
    nacm:default-deny-write;
    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 {
  nacm:default-deny-all;
  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; // FIXME: does this need to be mandatory?
    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.";
  }
}
3. Security Considerations

3.1. Support for Algorithms

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.

3.2. No Support for CRMF

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, a backwards compatible solution can be defined at that time.

3.3. Access to Data Nodes

The YANG module in this document defines "grouping" statements that 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 module 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 by the grouping statements that are writable/creatable/deletable (i.e., config true, which is the default). Some of 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:

*: All of the data nodes defined by all the groupings are considered sensitive to write operations. For instance, the modification of a public key or a certificate can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been applied to all the data nodes defined by all the groupings.

Some of the readable data nodes in the 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:

/private-key: The "private-key" node defined in the "asymmetric-key-pair-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it here.

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:

*: All of the "action" statements defined by groupings SHOULD only be executed by authorized users. For this reason, the NACM extension "default-deny-all" has been applied to all of them. Note that NACM uses "default-deny-all" to protect "RPC" and "action" statements; it does not define, e.g., an extension called "default-deny-execute".
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.

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:

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 following registration is requested:

name: ietf-crypto-types
prefix: ct
reference: RFC XXXX

5. References

5.1. Normative References


5.2. Informative References


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.

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

A.2. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action in use with the NETCONF protocol.
A.3. The "certificate-expiration" Notification

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

- renamed base identity ‘asymmetric-key-encryption-algorithm’ to ‘asymmetric-key-algorithm’.
- added new ‘asymmetric-key-algorithm’ identities for secp192r1, secp224r1, secp256r1, secp384r1, and secp521r1.
- for all -cbc and -ctr identities, renamed base identity ‘symmetric-key-encryption-algorithm’ to ‘encryption-algorithm’.
- 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.
- for all the ‘signature-algorithm’ based identities, renamed from ‘rsa-*’ to ‘rsassa-*’.
- removed all of the "x509v3-" prefixed ‘signature-algorithm’ based identities.
- added ‘key-exchange-algorithm’ based identities for ‘rsaes-oaep’ and ‘rsaes-pkcs1-v1_5’.
- renamed typedef ‘symmetric-key-encryption-algorithm-ref’ to ‘symmetric-key-algorithm-ref’.
- renamed typedef ‘asymmetric-key-encryption-algorithm-ref’ to ‘asymmetric-key-algorithm-ref’.
- added typedef ‘encryption-and-mac-algorithm-ref’.
- Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.5. 03 to 04

- ran YANG module through formatter.
B.6.  04 to 05
  o fixed broken symlink causing reformatted YANG module to not show.

B.7.  05 to 06
  o Added NACM annotations.
  o Updated Security Considerations section.
  o Added 'asymmetric-key-pair-with-cert-grouping' grouping.
  o Removed text from 'permanently-hidden' enum regarding such keys
    not being backed up or restored.
  o Updated the boilerplate text in module-level "description"
    statement to match copyeditor convention.
  o Added an explanation to the 'public-key-grouping' and 'asymmetric-
    key-pair-grouping' statements as for why the nodes are not
    mandatory (e.g., because they may exist only in <operational>).
  o Added 'must' expressions to the 'public-key-grouping' and
    'asymmetric-key-pair-grouping' statements ensuring sibling nodes
    are either all exist or do not all exist.
  o Added an explanation to the 'permanently-hidden' that the value
    cannot be configured directly by clients and servers MUST fail any
    attempt to do so.
  o Added 'trust-anchor-certs-grouping' and 'end-entity-certs-
    grouping' (the plural form of existing groupings).
  o Now states that keys created in <operational> by the *-hidden-key
    actions are bound to the lifetime of the parent 'config true'
    node, and that subsequent invocations of either action results in
    a failure.

B.8.  06 to 07
  o Added clarifications that implementations SHOULD assert that
    configured certificates contain the matching public key.
  o Replaced the 'generate-hidden-key' and 'install-hidden-key'
    actions with special 'crypt-hash' -like input/output values.
B.9.  07 to 08

- Removed the 'generate-key and 'hidden-key' features.
- Added grouping symmetric-key-grouping
- Modified 'asymmetric-key-pair-grouping' to have a 'choice' statement for the keystone module to augment into, as well as replacing the 'union' with leafs (having different NACM settings).

B.10.  08 to 09

- Converting algorithm from identities to enumerations.

B.11.  09 to 10

- All of the below changes are to the algorithm enumerations defined in ietf-crypto-types.
- Add in support for key exchange over x.25519 and x.448 based on RFC 8418.
- Add in SHAKE-128, SHAKE-224, SHAKE-256, SHAKE-384 and SHAKE 512
- Revise/add in enum of signature algorithm for x25519 and x448
- Add in des3-cbc-shal for IPSec
- Add in shal-des3-kd for IPSec
- Add in definit for rc4-hmac and rc4-hmac-exp. These two algorithms have been deprecated in RFC 8429. But some existing draft in i2nsf may still want to use them.
- Add x25519 and x448 curve for asymmetric algorithms
- Add signature algorithms ed25519, ed25519-cts, ed25519ph
- Add signature algorithms ed448, ed448ph
- Add in rsa-sha2-256 and rsa-sha2-512 for SSH protocols (rfc8332)

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Martin Bjorklund, Nick Hancock, Balazs Kovacs, Juergen Schoenwaelder, 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
Abstract

This document defines a YANG 1.1 module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when 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:

- "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:

- "2019-07-02" --> the publication date of this draft

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

- Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2020.

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 ............................................ 12
      3.2.1. A Keystore Instance ............................... 12
      3.2.2. The "generate-symmetric-key" RPC ................ 14
      3.2.3. Notable Keystore Groupings ......................... 14
      3.3. YANG Module ........................................... 18
3. Security Considerations ........................................ 27
3. TIANA Considerations ........................................... 28
   5.1. The IETF XML Registry .................................... 29
   5.2. The YANG Module Names Registry ........................... 29
6. References ...................................................... 29
   6.1. Normative References .................................... 29
   6.2. Informative References .................................. 30
Appendix A. Change Log ............................................. 31
   A.1. 00 to 01 ................................................. 31
   A.2. 01 to 02 ................................................. 31
   A.3. 02 to 03 ................................................. 31
   A.4. 03 to 04 ................................................. 31
   A.5. 04 to 05 ................................................. 32
   A.6. 05 to 06 ................................................. 32
   A.7. 06 to 07 ................................................. 32
1. Introduction

This document defines a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

The "ietf-keystore" module defines many "grouping" statements intended for use by other modules that may import it. For instance, there are groupings that defined enabling a key to be either configured locally (within the defining data model) or be a reference to a key in the keystore.

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 hides the secret key values. To support such hardware, symmetric keys may have the value "hidden-key" and asymmetric keys may have the value "hidden-private-key". While how such keys are created or destroyed is outside the scope of this document, the keystore can contain entries for such keys, enabling them to be reference by other configuration elements.

This document in compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, 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>.

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-t
                +++-rw public-key                binary
                +++-rw (private-key-type)
                    |  (private-key)
                        |      +++-rw private-key?        binary
                        |  (hidden-private-key)
                            |      +++-rw hidden-private-key? empty
                        |  (encrypted-private-key)
                            |      +++-rw encrypted-private-key
                                +++-rw (key-type)
                                    |  (symmetric-key-ref)
                                        |      +++-rw symmetric-key-ref? leafref
                                            (keystore-supported)?
                                        |  (asymmetric-key-ref)
                                            |      +++-rw asymmetric-key-ref? leafref
                                                (keystore-supported)?
                                            +++-rw value?        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

        +++-rw symmetric-keys
            +++-rw symmetric-key* [name]
                +++-rw name                string
                +++-rw algorithm
                    encryption-algorithm-t
                +++-rw (key-type)
Internet-Draft      A YANG Data Model for a Keystore           July 2019

+-:(key)
  |  +-:(key)
  |    |  +-rw key?             binary
  |    |  +-rw hidden-key?      empty
  |    |  +-:(encrypted-key)
  |    |    +-rw encrypted-key
  |    |    |    +-rw (key-type)
  |    |    |    |    +-:(symmetric-key-ref)
  |    |    |    |    |    +-rw symmetric-key-ref?    leafref
  |    |    |    |    |    |    (keystore-supported)?
  |    |    |    |    +-:(asymmetric-key-ref)
  |    |    |    |    |    +-rw asymmetric-key-ref?   leafref
  |    |    |    |    |    |    (keystore-supported)?
  |    |    |    +-rw value?                      binary

rpcs:
  +-x generate-symmetric-key
    +-w input
      |    |    |    +-w algorithm       ct:encryption-algorithm-t
      |    |    |    +-w encrypt-with!
      |    |    |    +-w (key-type)
      |    |    |    |    +-:(symmetric-key-ref)
      |    |    |    |    |    +-w symmetric-key-ref?    leafref
      |    |    |    |    |    |    (keystore-supported)?
      |    |    |    |    +-:(asymmetric-key-ref)
      |    |    |    |    |    +-w asymmetric-key-ref?   leafref
      |    |    |    |    |    |    (keystore-supported)?
    +-ro output
      |    |    +-ro algorithm              encryption-algorithm-t
      |    |    +-ro (key-type)
      |    |         |    |    +-ro key?             binary
      |    |         |    |    +-ro hidden-key?      empty
      |    |         |    |    +-:(encrypted-key)
      |    |         |    |    |    +-ro encrypted-key
      |    |         |    |    |    |    +-ro (key-type)
      |    |         |    |    |    |    |    +-ro symmetric-key-ref?    leafref
      |    |         |    |    |    |    |    |    (keystore-supported)?
      |    |         |    |    |    |    |    +-:(asymmetric-key-ref)
      |    |         |    |    |    |    |    +-ro asymmetric-key-ref?   leafref
      |    |         |    |    |    |    |    |    (keystore-supported)?
      |    |         |    |    |    +-ro value?                      binary

  +-x generate-asymmetric-key
    +-w input
      |    |    |    +-w algorithm       ct:asymmetric-key-algorithm-t
      |    |    |    +-w encrypt-with!
grouping key-reference-type-grouping
  +-- (key-type)
  |    +--:(symmetric-key-ref)
  |      +-- symmetric-key-ref? leafref
  |          (keystore-supported)?
  |      +--:(asymmetric-key-ref)
  |          +-- asymmetric-key-ref? leafref
  |          (keystore-supported)?
  +-- ro output
  +-- ro algorithm
  |    +-- ro public-key binary
  +-- ro (private-key-type)
  |    +-- ro (private-key)
  |      +-- ro private-key? binary
  |      +-- ro hidden-private-key? empty
  |      +-- ro encrypted-private-key
  |          +-- ro (key-type)
  |          |    +-- ro symmetric-key-ref? leafref
  |          |          (keystore-supported)?
  |          |      +-- ro asymmetric-key-ref? leafref
  |          |          (keystore-supported)?
  |          +-- ro value? binary

grouping encrypted-value-grouping
  +-- (key-type)
  |    +--:(symmetric-key-ref)
  |      +-- symmetric-key-ref? -> /keystore/symmetric-keys/symmetric-key/name
  |          (keystore-supported)?
  |      +--:(asymmetric-key-ref)
  |      +-- asymmetric-key-ref? -> /keystore/asymmetric-keys/asymmetric-key/name
  |          (keystore-supported)?
  +-- ro value? binary
grouping symmetric-key-grouping
--- algorithm encryption-algorithm-t
--- (key-type)
  ---:(key)
   |  --- key? binary
  ---:(hidden-key)
   |  --- hidden-key? empty
  ---:(encrypted-key)
   --- encrypted-key
    --- (key-type)
     |     ---:(symmetric-key-ref)
     |     |     --- symmetric-key-ref? leafref
     |     |     (keystore-supported)?
     |     ---:(asymmetric-key-ref)
     |     |     --- asymmetric-key-ref? leafref
     |     |     (keystore-supported)?
     --- value? binary

grouping asymmetric-key-pair-grouping
--- algorithm asymmetric-key-algorithm-t
--- public-key binary
--- (private-key-type)
  ---:(private-key)
   |   --- private-key? binary
  ---:(hidden-private-key)
   |   --- hidden-private-key? empty
  ---:(encrypted-private-key)
   --- encrypted-private-key
    --- (key-type)
     |     ---:(symmetric-key-ref)
     |     |     --- symmetric-key-ref? leafref
     |     |     (keystore-supported)?
     |     ---:(asymmetric-key-ref)
     |     |     --- asymmetric-key-ref? leafref
     |     |     (keystore-supported)?
     --- value? binary

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

grouping keystore-grouping
  +-- symmetric-keys
    |  +-- symmetric-key* [name]
    |     +-- name?                  string
    |     +-- algorithm              encryption-algorithm-t
    |     +-- (key-type)
    |     |  +--:(key)
    |     |     |  +-- key?                        binary
    |     |  +--:(hidden-key)
    |     |     |  +-- hidden-key?                 empty
    |     |  +--:(encrypted-key)
    |     |     |  +-- encrypted-key
    |     |     |     |  +-- (key-type)
    |     |     |     |  |  +--:(symmetric-key-ref)
3.2. Example Usage

3.2.1. A Keystore Instance

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) ===========

```xml
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
    xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
    or:origin="or:intended">
  <!-- Asymmetric Keys -->
  <asymmetric-keys>
    <asymmetric-key>
      <name>ex-rsa-key</name>
      <algorithm>rsa2048</algorithm>
      <public-key>base64encodedvalue==</public-key>
      <private-key>base64encodedvalue==</private-key>
      <certificates>
        <certificate>
          <name>ex-rsa-cert</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>tls-ec-key</name>
      <algorithm>secp256r1</algorithm>
      <public-key>base64encodedvalue==</public-key>
      <private-key>base64encodedvalue==</private-key>
      <certificates>
        <certificate>
          <name>ex-ec-cert</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```
<name>tls-ec-cert</name>
<cert>base64encodedvalue==</cert>
</certificate>
</certificates>
</asymmetric-key>

<asymmetric-key>
<name>tpm-protected-key</name>
<algorithm>rsa2048</algorithm>
<public-key>base64encodedvalue==</public-key>
<hidden-private-key/>
<certificates>
  <certificate>
    <name>builtin-idevid-cert</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
  <certificate>
    <name>my-idevid-cert</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
</certificates>
</asymmetric-key>

<asymmetric-key>
<name>encrypted-key</name>
<algorithm>secp256r1</algorithm>
<public-key>base64encodedvalue==</public-key>
<encrypted-private-key>
  <symmetric-key-ref>operators-encrypted-key</symmetric-key-ref>
  <value>base64encodedvalue==</value>
</encrypted-private-key>
</asymmetric-key>
</asymmetric-keys>

<!-- Symmetric Keys -->
<symmetric-keys>

</symmetric-keys>
3.2.2. The "generate-symmetric-key" RPC

The following example illustrates the "generate-symmetric-key" RPC. The key being referenced is defined in the keystore example above.

```xml
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <generate-symmetric-key
       xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    <algorithm>aes-256-cbc</algorithm>
    <encrypt-with>
      <asymmetric-key-ref>tpm-protected-key</asymmetric-key-ref>
    </encrypt-with>
  </generate-symmetric-key>
</rpc>
```

Following is the complimentary RPC-reply.

```
<rpc-reply message-id="101"
            xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
            xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-keystore">
  <!----<data> yanglint validation fails -->
  <ks:algorithm>aes-256-cbc</ks:algorithm>
  <ks:encrypted-key>
    <ks:asymmetric-key-ref>tpm-protected-key</ks:asymmetric-key-ref>
    <ks:value>base64encodedvalue==</ks:value>
  </ks:encrypted-key>
  <!----<data> yanglint validation fails -->
</rpc-reply>
```

3.2.3. Notable Keystore Groupings

The following non-normative module is used by subsequent examples to illustrate groupings defined in the ietf-crypto-types module.

```yang
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>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-usage>
<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>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>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)</name>
  <keystore-reference>
    <asymmetric-key>ex-rsa-key</asymmetric-key>
    <certificate>ex-rsa-cert</certificate>
  </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@2019-07-02.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."

  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

  This version of this YANG module is part of RFC XXXX

Watsen Expires January 3, 2020 [Page 18]
Internet-Draft      A YANG Data Model for a Keystore           July 2019

{https://www.rfc-editor.org/info/rfcXXXX}; see the RFC itself for full legal notices.;

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

revision 2019-07-02 {
  description
    "Initial version";
  reference
    "RFC VVVV: A YANG Data Model for a Keystore";
}

/****************/
/*   Features   */
/****************/

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

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

feature key-generation {
  description
    "Indicates that the server supports the actions related to the life cycling keys in <operational>. To be used by configuration, keys in <operational> must be copied to <running>.";
}

/****************/
/*   Typedefs   */
****************/

typedef asymmetric-key-ref {
  type leafref {
    path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
grouping key-reference-type-grouping {
    description "A reusable grouping for a choice for the type of key referenced in the keystore.";
    choice key-type {
        mandatory true;
        description "A choice between a reference to a symmetric or asymmetric key in the keystore.";
        leaf symmetric-key-ref {
            if-feature "keystore-supported";
            type leafref {
                path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key/" + "ks:name";
            }
            description "Identifies a symmetric key used to encrypt this key.";
        }
        leaf asymmetric-key-ref {
            if-feature "keystore-supported";
            type leafref {
                path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key/" + "ks:name";
            }
            description "Identifies an asymmetric key used to encrypt this key.";
        }
    }
}

grouping encrypted-value-grouping {
    description "A reusable grouping for a value that has been encrypted by a symmetric or asymmetric key in the keystore.";
    uses "key-reference-type-grouping";
    leaf value {
        + "/ks:name";
    }
    description "This typedef enables modules to easily define a reference to an asymmetric key stored in the keystore.";
}
type binary;
description
"The private key, encrypted using the specified symmetric or asymmetric key.";
}
}

grouping symmetric-key-grouping {

description
"This grouping is identical to the one in ietf-crypt-types except that it adds a couple case statements enabling the key value to be encrypted by a symmetric or an asymmetric key known to the keystore.";
uses ct:symmetric-key-grouping {
  augment "key-type" {
    description
    "Augments a new 'case' statement into the 'choice' statement defined by the ietf-crypto-types module.";
    container encrypted-key {
      description
      "A container for the encrypted symmetric key value.";
      uses encrypted-value-grouping;
    }
  }
}
}


grouping asymmetric-key-pair-grouping {

description
"This grouping is identical to the one in ietf-crypt-types except that it adds a couple case statements enabling the key value to be encrypted by a symmetric or an asymmetric key known to the keystore.";
uses ct:asymmetric-key-pair-grouping {
  augment "private-key-type" {
    description
    "Augments a new 'case' statement into the 'choice' statement defined by the ietf-crypto-types module.";
    container encrypted-private-key {
      description
      "A container for the encrypted asymmetric private key value.";
      uses encrypted-value-grouping;
    }
  }
}
}
grouping asymmetric-key-pair-with-cert-grouping {

description
"This grouping is identical to the one in ietf-crypt-types except that it adds a couple case statements enabling the key value to be encrypted by a symmetric or an asymmetric key known to the keystore."

uses ct:asymmetric-key-pair-with-cert-grouping {

augment "private-key-type" {

description
"Augments a new 'case' statement into the 'choice' statement defined by the ietf-crypto-types module."

container encrypted-private-key {

description
"A container for the encrypted asymmetric private key value."

uses encrypted-value-grouping;
}
}
}
}
}

}
leaf certificate {
  type leafref {
    path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key[ks:
        + "name = current()]/../asymmetric-key]/ks:certificates"
        + "/ks:certificate/ks:name";
  }
  must '../asymmetric-key';
  description
    "A reference to a specific certificate of the asymmetric key in the keystore.";
}

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-definitions-supported";
      container local-definition {
        description
          "Container to hold the local key definition.";
        uses 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-definitions-supported";
      container local-definition {
        description
        "Container to hold the local key definition.";
        uses asymmetric-key-pair-with-certs-grouping;
      }
    }
    case keystore {
      if-feature "keystore-supported";
      leaf keystore-reference {
        type ks:asymmetric-key-ref;
        description
        "A reference to an asymmetric-key (and all of its
        associated certificates) 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-definitions-supported";
      container local-definition {
        description
        "Container to hold the local key definition.";
        uses asymmetric-key-pair-with-cert-grouping;
      }
    }
    case keystore {
      
    }
  }
}
if-feature "keystore-supported";
container keystore-reference {
    uses asymmetric-key-certificate-ref-grouping;
    description
    "A reference to a specific certificate (and its
    associated private key) in the keystore.";
}

description
"A choice between an inlined definition and a definition
that exists in the keystore."
}

grouping keystore-grouping {
    description
    "Grouping definition enables use in other contexts. If ever
done, implementations SHOULD augment new ‘case’ statements
into local-or-keystore ‘choice’ statements to supply leafrefs
to the new location.";
    container asymmetric-keys {
        description
        "A list of asymmetric keys.";
        list asymmetric-key {
            key "name";
            description
            "An asymmetric key.";
            leaf name {
                type string;
                description
                "An arbitrary name for the asymmetric key.";
            }
            uses ks:asymmetric-key-pair-with-certs-grouping;
        }
    }
    container symmetric-keys {
        description
        "A list of symmetric keys.";
        list symmetric-key {
            key "name";
            description
            "A symmetric key.";
            leaf name {
                type string;
                description
                "An arbitrary name for the symmetric key.";
            }
            uses ks:symmetric-key-grouping;
        }
    }
}
container keystore {
  nacm:default-deny-write;
  description
    "The keystore contains a list of keys.";
  uses keystore-grouping;
}

rpc generate-symmetric-key {
  //nacm:default-deny-all;
  description
    "Requests the device to generate an symmetric key using
    the specified key algorithm, optionally encrypted using
    a key in the keystore. The output is this RPC can be
    used as input to a subsequent configuration request.";
  input {
    leaf algorithm {
      type ct:encryption-algorithm-t;
      mandatory true;
      description
        "The algorithm to be used when generating the key.";
      reference
        "RFC CCCC: Common YANG Data Types for Cryptography";
    }
    container encrypt-with {
      presence
        "Indicates that the key should be encrypted using
        the specified symmetric or asymmetric key. If not
        specified, then the private key is not encrypted
        when returned.";
      description
        "A container for the 'key-type' choice.";
      uses key-reference-type-grouping;
    }
  }
  output {
    uses ks:symmetric-key-grouping;
  }
} // end generate-symmetric-key
rpc generate-asymmetric-key {
  //nacm:default-deny-all;
  description
  "Requests the device to generate an asymmetric key using
  the specified key algorithm, optionally encrypted using
  a key in the keystore. The output is this RPC can be
  used as input to a subsequent configuration request.";
  input {
    leaf algorithm {
      type ct:asymmetric-key-algorithm-t;
      mandatory true;
      description
      "The algorithm to be used when generating the key.";
      reference
      "RFC CCCC: Common YANG Data Types for Cryptography";
    }
    container encrypt-with {
      presence
      "Indicates that the key should be encrypted using
      the specified symmetric or asymmetric key. If not
      specified, then the private key is not encrypted
      when returned.";
      description
      "A container for the 'key-type' choice.";
      uses key-reference-type-grouping;
    }
  }
  output {
    uses ks:asymmetric-key-pair-grouping;
  }
} // end generate-asymmetric-key

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.

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:

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 following registration is requested:

name: ietf-keystore
prefix: ks
reference: RFC VVVV

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]


6.2. Informative References


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.

A.9. 08 to 09
   o Added a 'description' statement to the 'must' in the /keystore/asymmetric-key node explaining that the descendent values may exist in <operational> only, and that implementation MUST assert that the values are either configured or that they exist in <operational>.
Internet-Draft      A YANG Data Model for a Keystore           July 2019

o Copied above 'must' statement (and description) into the local-or-
keystore-asymmetric-key-grouping, local-or-keystore-asymmetric-
key-with-certs-grouping, and local-or-keystore-end-entity-cert-
with-key-grouping statements.

A.10. 09 to 10

o Updated draft title to match new truststore draft title

o Moved everything under a top-level 'grouping' to enable use in
other contexts.

o Renamed feature from 'local-keys-supported' to 'local-definitions-
supported' (same name used in truststore)

o Removed the either-all-or-none 'must' expressions for the key’s
3-tuple values (since the values are now 'mandatory true' in
crypto-types)

o Example updated to reflect 'mandatory true' change in crypto-types
draft

A.11. 10 to 11

o Replaced typedef asymmetric-key-certificate-ref with grouping
asymmetric-key-certificate-ref-grouping.

o Added feature feature 'key-generation'.

o Cloned groupings symmetric-key-grouping, asymmetric-key-pair-
grouping, asymmetric-key-pair-with-cert-grouping, and asymmetric-
key-pair-with-certs-grouping from crypto-keys, augmenting into
each new case statements for values that have been encrypted by
other keys in the keystore. Refactored keystore model to use
these groupings.

o Added new 'symmetric-keys' lists, as a sibling to the existing
'asymmetric-keys' list.

o Added RPCs (not actions) 'generate-symmetric-key' and 'generate-
asymmetric-key' to *return* a (potentially encrypted) key.

A.12. 11 to 12

o Updated to reflect crypto-type’s draft using enumerations over
identities.
o Added examples for the 'generate-symmetric-key' and 'generate-asymmetric-key' RPCs.

o Updated the Introduction section.

Acknowledgements

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

Author’s Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net
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:

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

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

- "XXXX" --> the assigned RFC value for this draft
- "AAAA" --> the assigned RFC value for I-D.ietf-netconf-tcp-client-server
- "YYYY" --> the assigned RFC value for I-D.ietf-netconf-ssh-client-server
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].
Internet-Draft      NETCONF Client and Server Models           July 2019

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, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon 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.

This tree diagram only shows the nodes defined in this module; it does show the nodes defined by "grouping" statements used by this module.

Please see Appendix A.1 for a tree diagram that illustrates what the module looks like with all the "grouping" statements expanded.

module: ietf-netconf-client
  +--rw netconf-client
    +--u netconf-client-grouping

  grouping netconf-client-grouping
    +-- initiate! {ssh-initiate or tls-initiate}?
      |  +-- netconf-server* [name]
      |    +-- name? string
      |    +-- endpoints
      |      +-- endpoint* [name]
      |        +-- name? string
      |        +-- (transport)
      |          +--:(ssh) {ssh-initiate}?
      |            |  +-- ssh
      |            |    +-- tcp-client-parameters
      |            |    +-- u tcpc:tcp-client-grouping
      |            |    +-- ssh-client-parameters
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 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].
<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>
          <tcp-client-parameters>
            <remote-address>corp-fw1.example.com</remote-address>
            <keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </keepalives>
          </tcp-client-parameters>
          <ssh-client-parameters>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <local-definition>
                  <algorithm>rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                </local-definition>
              </public-key>
            </client-identity>
            <server-authentication>
              <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
              <server-certs>explicitly-trusted-server-certs</server-certs>
            </server-authentication>
            <keepalives>
              <max-wait>30</max-wait>
              <max-attempts>3</max-attempts>
            </keepalives>
          </ssh-client-parameters>
        </ssh>
      </endpoint>
      <endpoint>
        <name>corp-fw2.example.com</name>
        <ssh>
          <tcp-client-parameters>
            <remote-address>corp-fw2.example.com</remote-address>
            <keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </keepalives>
          </tcp-client-parameters>
          <ssh-client-parameters>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <local-definition>
                  <algorithm>rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                </local-definition>
              </public-key>
            </client-identity>
            <server-authentication>
              <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
              <server-certs>explicitly-trusted-server-certs</server-certs>
            </server-authentication>
            <keepalives>
              <max-wait>30</max-wait>
              <max-attempts>3</max-attempts>
            </keepalives>
          </ssh-client-parameters>
        </ssh>
      </endpoint>
    </endpoints>
  </netconf-server>
</initiate>
</netconf-client>
<tcp-client-parameters>
  <remote-address>corp-fw2.example.com</remote-address>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-client-parameters>

<ssh-client-parameters>
  <client-identity>
    <username>foobar</username>
    <public-key>
      <local-definition>
        <algorithm>rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
      </local-definition>
    </public-key>
  </client-identity>
  <server-authentication>
    <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
    <server-certs>explicitly-trusted-server-certs</server-certs>
  </server-authentication>
  <keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </keepalives>
</ssh-client-parameters>

<connection-type>
  <persistent/>
</connection-type>

<reconnect-strategy>
  <start-with>last-connected</start-with>
</reconnect-strategy>
</netconf-server>
3.3. YANG Module

This YANG module has normative references to [RFC6242], [RFC6991], [RFC7589], [RFC8071], [I-D.watsen-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@2019-07-02.yang"
module ietf-netconf-client {  
yang-version 1.1;  
prefix ncc;  
import ietf-yang-types {  
  prefix yang;  
  reference  
  "RFC 6991: Common YANG Data Types";  
}  
import ietf-tcp-client {  
  prefix tcpc;  
  reference  
}

Watsen Expired January 3, 2020 [Page 8]
"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 2019-07-02; // stable grouping definitions
  reference
    "RFC BBBB: YANG Groupings for SSH Clients and SSH Servers";
}

import ietf-tls-client {
  prefix tlsc;
  revision-date 2019-07-02; // stable grouping definitions
  reference
    "RFC CCCC: 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.

  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

  This version of this YANG module is part of RFC XXXX
   (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
   itself for full legal notices.";}
The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

revision 2019-07-02 {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Client and Server Models";
}

// Features

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 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.";
// Groupings

grouping netconf-client-grouping {
    description "Top-level grouping for NETCONF client configuration.";
    container initiate {
        if-feature "ssh-initiate or tls-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."
    container tcp-client-parameters {
        description
        "A wrapper around the TCP client parameters
        to avoid name collisions."
        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."
            }
        }
    }
}

container ssh-client-parameters {
    description
    "A wrapper around the SSH client parameters to
    avoid name collisions."
    uses sshc:ssh-client-grouping;
}
}
}

case tls {
    if-feature "tls-initiate";
    container tls {
        description
        "Specifies IP and TLS specific configuration
        for the connection."
        container tcp-client-parameters {
            description
            "A wrapper around the TCP client parameters
            to avoid name collisions."
            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."
                }
            }
        }
    }
}
container tls-client-parameters {
    must "client-identity" {
        description
        "NETCONF/TLS clients MUST pass some authentication credentials.";
    }
    description
    "A wrapper around the TLS client parameters to avoid name collisions.";
    uses tlsc:tls-client-grouping;
}

} // 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 the NETCONF server, 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.

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

The NETCONF client should close the underlying
TCP connection upon completing planned activities.

In the case that the previous connection is still
active, establishing a new connection is NOT
RECOMMENDED.;
leaf period {
    type uint16;
    units "minutes";
    default "60";
    description
    "Duration of time between periodic connections.";
}
leaf anchor-time {
    type yang:date-and-time {
        // constrained to minute-level granularity
        pattern '\(\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:[\d\-]+\)'
        + '(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 then 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).";
    }
}
container listen {
  if-feature "ssh-listen or tls-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.";
          container tcp-server-parameters {
            description
            "A wrapper around the TCP server parameters
            to avoid name collisions.";
            uses tcps:tcp-server-grouping {
              refine "local-port" {
                default "4334";
              }
            }
          }
        }
      }
    }
  }
} // netconf-server
} // initiate
description
"The NETCONF client will listen on the IANA-assigned well-known port for 'netconf-ch-ssh' (4334) if no value is specified.";
}
}
container ssh-client-parameters {
  description
  "A wrapper around the SSH client parameters to avoid name collisions.";
  uses sshc:ssh-client-grouping;
}
}
}
case tls {
  if-feature "tls-listen";
  container tls {
    description
    "TLS-specific listening configuration for inbound connections.";
    container tcp-server-parameters {
      description
      "A wrapper around the TCP server parameters to avoid name collisions.";
      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.";
        }
      }
    }
    container tls-client-parameters {
      must "client-identity" {
        description
        "NETCONF/TLS clients MUST pass some authentication credentials.";
      }
      description
      "A wrapper around the TLS client parameters to avoid name collisions.";
      uses tlsc:tls-client-grouping;
    }
  }
}
4. The NETCONF Server Model

The NETCONF server model presented in this section supports both listening for connections as well as initiating call-home connections, using either the SSH and TLS transport protocols.

YANG feature statements are used to enable implementations to advertise which potentially uncommon 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.

This tree diagram only shows the nodes defined in this module; it does show the nodes defined by "grouping" statements used by this module.

Please see Appendix A.2 for a tree diagram that illustrates what the module looks like with all the "grouping" statements expanded.
+++:(ssh) {ssh-listen}?
+++ ssh
   +++ tcp-server-parameters
      |   +++-u tcps:tcp-server-grouping
      +++ ssh-server-parameters
         +++-u sshs:ssh-server-grouping
+++:(tls) {tls-listen}?
+++ tls
   +++ tcp-server-parameters
      |   +++-u tcps:tcp-server-grouping
      +++ tls-server-parameters
         +++-u tlss:tls-server-grouping
+++ call-home! {ssh-call-home or tls-call-home}?
+++ netconf-client* [name]
   +++ name?  string
+++ endpoints
   +++ endpoint* [name]
      +++ name?  string
      +++ (transport)
         +++:(ssh) {ssh-call-home}?
            +++ ssh
               +++ tcp-client-parameters
                  |   +++-u tcpc:tcp-client-grouping
                  +++ ssh-server-parameters
                     +++-u sshs:ssh-server-grouping
            +++:(tls) {tls-call-home}?
               +++ tls
                  +++ tcp-client-parameters
                     |   +++-u tcpc:tcp-client-grouping
                  +++ tls-server-parameters
                     +++-u tlss:tls-server-grouping
+++ connection-type
   +++ (connection-type)
      +++:(persistent-connection)
         |   +++ persistent!
      +++:(periodic-connection)
         +++ periodic!
            +++ period?  uint16
            +++ anchor-time?  yang:date-and-time
            +++ idle-timeout?  uint16
+++ reconnect-strategy
   +++ start-with?  enumeration
   +++ max-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 2 of [I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

======= NOTE: ‘\’ 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>
      <tcp-server-parameters>
        <local-address>192.0.2.7</local-address>
      </tcp-server-parameters>
      <ssh-server-parameters>
        <server-identity>
          <host-key>
            <name>deployment-specific-certificate</name>
            <public-key>
              <local-definition>
                <algorithm>rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
              </local-definition>
            </public-key>
          </host-key>
        </server-identity>
        <client-authentication>
          <supported-authentication-methods>
            <publickey/>
          </supported-authentication-methods>
        </client-authentication>
      </ssh-server-parameters>
    </ssh>
  </endpoint>
  <endpoint> <!-- listening for TLS sessions -->
    <name>netconf/tls</name>
    <tls>
      <tcp-server-parameters>
        <local-address>192.0.2.7</local-address>
      </tcp-server-parameters>
      <tls-server-parameters>
        <server-identity>
          <host-key>
            <name>deployment-specific-certificate</name>
            <public-key>
              <local-definition>
                <algorithm>rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
              </local-definition>
            </public-key>
          </host-key>
        </server-identity>
        <client-authentication>
          <supported-authentication-methods>
            <publickey/>
          </supported-authentication-methods>
        </client-authentication>
      </tls-server-parameters>
    </tls>
  </endpoint>
</listen>

</netconf-server>
<name>netconf/tls</name>
<tls>
  <tcp-server-parameters>
    <local-address>192.0.2.7</local-address>
  </tcp-server-parameters>
  <tls-server-parameters>
    <server-identity>
      <local-definition>
        <algorithm>rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </server-identity>
    <client-authentication>
      <required/>
      <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
      <client-certs>explicitly-trusted-client-certs</client-certs>
      <cert-maps>
        <cert-to-name>
          <id>1</id>
          <fingerprint>11:0A:05:11:00</fingerprint>
        </cert-to-name>
        <cert-to-name>
          <id>2</id>
          <fingerprint>B3:4F:A1:8C:54</fingerprint>
        </cert-to-name>
      </cert-maps>
    </client-authentication>
  </tls-server-parameters>
</tls>
</end-point>
</listen>

<!-- calling home to SSH and TLS based NETCONF clients -->
<!-- calling home -->
<netconf-client> <!-- SSH-based client -->
  <name>config-mgr</name>
  <endpoints>
    <endpoint>
      <name>east-data-center</name>
      <ssh>
        <tcp-client-parameters>
          <remote-address>east.config-mgr.example.com</remote-address>
        </tcp-client-parameters>
      </ssh>
    </endpoint>
  </endpoints>
</netconf-client>
<tcp-client-parameters>
  <remote-address>west.config-mgr.example.com</remote-address>
</tcp-client-parameters>

<ssh-server-parameters>
  <server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <local-definition>
          <algorithm>rsa2048</algorithm>
          <private-key>base64encodedvalue==</private-key>
          <public-key>base64encodedvalue==</public-key>
        </local-definition>
      </public-key>
    </host-key>
  </server-identity>
  <client-authentication>
    <supported-authentication-methods>
      <publickey/>
    </supported-authentication-methods>
    <client-auth-defined-elsewhere/>
  </client-authentication>
</ssh-server-parameters>

</ssh>
</endpoint>

<endpoint>
  <name>west-data-center</name>
  <ssh>
    <tcp-client-parameters>
      <remote-address>west.config-mgr.example.com</remote-address>
    </tcp-client-parameters>
    <ssh-server-parameters>
      <server-identity>
        <host-key>
          <name>deployment-specific-certificate</name>
          <public-key>
            <local-definition>
              <algorithm>rsa2048</algorithm>
              <private-key>base64encodedvalue==</private-key>
              <public-key>base64encodedvalue==</public-key>
            </local-definition>
          </public-key>
        </host-key>
      </server-identity>
      <client-authentication>
        <supported-authentication-methods>
          <publickey/>
        </supported-authentication-methods>
        <client-auth-defined-elsewhere/>
      </client-authentication>
    </ssh-server-parameters>
  </ssh>
</endpoint>
<client-auth-defined-elsewhere/>
</client-authentication>
</ssh>
</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>
<tcp-client-parameters>
<remote-address>east.analytics.example.com</remote-address>
<keepalives>
<idle-time>15</idle-time>
<max-probes>3</max-probes>
<probe-interval>30</probe-interval>
</keepalives>
</tcp-client-parameters>
<tls-server-parameters>
<server-identity>
<local-definition>
<algorithm>rsa2048</algorithm>
<private-key>base64encodedvalue==</private-key>
<public-key>base64encodedvalue==</public-key>
<cert>base64encodedvalue==</cert>
</local-definition>
</server-identity>
</client-authentication>
<required/>
<ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
<client-certs>explicitly-trusted-client-certs</client-certs>
<cert-maps>
<cert-to-name>
  <id>1</id>
  <fingerprint>11:0A:05:11:00</fingerprint>
  <map-type>x509c2n:sd-an</map-type>
</cert-to-name>
<cert-to-name>
  <id>2</id>
  <fingerprint>B3:4F:A1:8C:54</fingerprint>
  <map-type>x509c2n:sd-specified</map-type>
  <name>scooby-doo</name>
</cert-to-name>
</cert-maps>
</client-authentication>
<keepalives>
  <max-wait>30</max-wait>
  <max-attempts>3</max-attempts>
</keepalives>
</tls-server-parameters>
</tls>
</endpoints>
</connection-type>
</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].

<CODE BEGINS> file "ietf-netconf-server@2019-07-02.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;

  }
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 2019-07-02; // stable grouping definitions
  reference
    "RFC BBBB: YANG Groupings for SSH Clients and SSH Servers";
}

import ietf-tls-server {
  prefix tlss;
  revision-date 2019-07-02; // stable grouping definitions
  reference
    "RFC CCCC: 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.

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

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

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’,
‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document
are to be interpreted as described in BCP 14 (RFC 2119)
(RFC 8174) when, and only when, they appear in all
capitals, as shown here.”;

revision 2019-07-02 {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF Client and Server Models";
}

// Features

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

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";
}

grouping netconf-server-grouping {
    description
    "Top-level grouping for NETCONF server configuration.";
    container listen {
        if-feature "ssh-listen or tls-listen";
        presence
        "Enables server to listen for NETCONF client 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.";
        }
    }
}

// Groupings
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.";
      container tcp-server-parameters {
        description
        "A wrapper around the TCP client parameters
        to avoid name collisions.";
        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.";
          }
        }
      }
      container ssh-server-parameters {
        description
        "A wrapper around the SSH server parameters
        to avoid name collisions.";
        uses sshs:ssh-server-grouping;
      }
    }
  }
  case tls {
    if-feature "tls-listen";
    container tls {
      description
      "TLS-specific listening configuration for inbound
      connections.";
    }
  }
}
container tcp-server-parameters {
    description
    "A wrapper around the TCP client parameters to avoid name collisions.";
    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.";
        }
    }
}

container tls-server-parameters {
    description
    "A wrapper around the TLS server parameters to avoid name collisions.";
    uses tlss:tls-server-grouping {
        refine "client-authentication" {
            //must 'ca-certs or client-certs';
            description
            "NETCONF/TLS servers MUST validate client certificates.";
        }
        augment "client-authentication" {
            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 WWWW: NETCONF over TLS, Section 7";
            }
        }
    }
}
container call-home {
  if-feature "ssh-call-home or tls-call-home";
  presence
  "Enables the NETCONF server to initiate the underlying transport connection to NETCONF clients.";
  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.";
          container tcp-client-parameters {
            description
          }
        }
      }
    }
  }
}
"A wrapper around the TCP client parameters to avoid name collisions.;
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."
  }
}
}
container ssh-server-parameters {
  description
  "A wrapper around the SSH server parameters to avoid name collisions.";
  uses sshs:ssh-server-grouping;
}
}
}
case tls {
  if-feature "tls-call-home";
  container tls {
    description
    "Specifies TLS-specific call-home transport configuration.";
    container tcp-client-parameters {
      description
      "A wrapper around the TCP client parameters to avoid name collisions.";
      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."
        }
      }
    }
    }
  }
  container tls-server-parameters {
    description
    "A wrapper around the TLS server parameters to avoid name collisions.";
    uses tlss:tls-server-grouping {
      refine "client-authentication"
    }
  }
}
Watsen Expires January 3, 2020 [Page 32]
/* commented out since auth could be external
must 'ca-certs or client-certs'; */

description
"NETCONF/TLS servers MUST validate client
certificates."
}

augment "client-authentication" {

description
"Augments in the cert-to-name structure.";

container cert-maps {
	names 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 WWW: NETCONF over TLS, Section 7";
}
}
}

}
}

} // tls
}

} // 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 the NETCONF client, 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."

        This connection type increases resource utilization, albeit with increased delay in NETCONF client to NETCONF client interactions.

        The NETCONF client SHOULD gracefully close the connection using <close-session> upon completing planned activities. If the NETCONF session is not closed gracefully, the NETCONF server MUST immediately attempt to reestablish the connection.

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

        leaf period {
            type uint16;
            units "minutes";
            default "60";
            description "Duration of time between periodic connections.";
        }

        leaf anchor-time {
            type yang:date-and-time {
                // constrained to minute-level granularity
                pattern '\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...";
        }
    }
}

Watsen
Expires January 3, 2020

[Page 34]
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. Security Considerations

The YANG module defined in this document uses groupings defined in
[I-D.kwatsen-netconf-tcp-client-server],
[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 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.

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). Some of 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 of the subtrees or data nodes in the modules defined in this document need to be protected from write operations.

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 of the subtrees or data nodes in the modules defined in this document need to be protected from read operations.

Some of the RPC operations in the YANG modules 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:

The modules defined in this document do not define any ‘RPC’ or ‘action’ statements.

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:
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-netconf-client
prefix:       ncc
reference:    RFC XXXX

name:         ietf-netconf-server
prefix:       ncs
reference:    RFC XXXX

7. References

7.1. Normative References

[I-D.ietf-netconf-keystore]

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

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

[I-D.kwatsen-netconf-tcp-client-server]
7.2. Informative References

[I-D.ietf-netconf-trust-anchors]


Appendix A. Expanded Tree Diagrams

A.1. Expanded Tree Diagram for 'ietf-netconf-client'

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-client" module.

This tree diagram shows all the nodes defined in this module, including those defined by "grouping" statements used by this module.

Please see Section 3.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

======= NOTE: ‘\\’ line wrapping per BCP XX (RFC XXXX) =======
++-rw server-authentication
  ++-rw ca-certs?
    ts:certificates-ref
    (ts:x509-certificates)?
  ++-rw server-certs?
    ts:certificates-ref
    (ts:x509-certificates)?
  ++-rw hello-params
    (tls-client-hello-params-config)\ 
  
  ++-rw tls-versions
  |  ++-rw tls-version* identityref
  ++-rw cipher-suites
  |  ++-rw cipher-suite* identityref
  ++-rw keepalives!
    (tls-client-keepalives)?
  ++-rw max-wait? uint16
  ++-rw max-attempts? uint8

++-rw connection-type
  ++-rw (connection-type)
    |  ++-rw persistent!
  ++-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! {ssh-listen or tls-listen}? 
  ++-rw idle-timeout? uint16
  ++-rw endpoint* [name]
    ++-rw name string
  ++-rw (transport)
    |  ++-:(ssh) {ssh-listen}? 
    |    ++-rw ssh
    |    |  ++-rw tcp-server-parameters
    |    |    ++-rw local-address inet:ip-address
    |    |    ++-rw local-port? inet:port-number
    |    |    ++-rw keepalives! {keepalives-supported}?
    |    |    |  ++-rw idle-time uint16
    |    |    |  ++-rw max-probes uint16
    |    |    |  ++-rw probe-interval uint16
    |    ++-rw ssh-client-parameters
    |    |  ++-rw client-identity
    |    |    |  ++-rw username? string
The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-netconf-server" module.

This tree diagram shows all the nodes defined in this module, including those defined by "grouping" statements used by this module.

Please see Section 4.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

========== NOTE: '\\' line wrapping per BCP XX (RFC XXXX) ===========
| +--rw keepalives! {keepalives-supported}? |
| +--rw idle-time  uint16  |
| +--rw max-probes  uint16  |
| +--rw probe-interval  uint16  |
| +--rw ssh-server-parameters  |
| +--rw server-identity  |
| +--rw host-key* [name]  |
| +--rw name  string  |
| +--(public-key)  |
| +--rw public-key  |
| +--rw (local-or-keystore)  |
| +--:(local)  |
| (-supported)?  |
| +--rw local-definition  |
| +--rw algorithm  |
| +-- asymmetric-key-t  |
| +--rw public-key  |
| +-- binary  |
| +--rw (private-key-type)  |
| +--:(private-key)  |
| +--rw private-k  |
| +-- hidden-key?  |
| +-- binary  |
| +--:(hidden-private-key)  |
| +--:(encrypted-private-key)  |
| +-- encrypted-key  |
| +--rw encrypted  |
| +-- key-type  |
| +--:(symmetric-key-ref)  |
| +--:(asymmetric-key-ref)  |
| +--rw symmetric-key-ref?  |
| +--:(keystore-supported)?  |
| +--:(asymmetric-key-ref)  |
| +--:(symmetric-key-ref)  |
---rw idle-time  uint16
  ---rw max-probes  uint16
  ---rw probe-interval  uint16

++-rw ssh-server-parameters
  ++-rw server-identity
    ++-rw host-key* [name]
      ++-rw name  string
      ++-rw (host-key-type)
        ++-:(public-key)
          ++-rw public-key

++-rw (local-or-keystore)
  ++-:(local)
    (local-definition-supported)?

++-rw local-definition
++-rw algorithm
  asymmetrical

++-rw public-key
  binary

++-rw (private-key)
++-:(private-key)
++-:(private-key-type)
++-:(private-key)
++-:(hidden-private-key)
++-:(encrypted-private-key)
++-rw encrypted-private-key
  ++-:(key-type)
    ++-:(symmetric-key-ref)
      ++-rw symmetric-key-ref?
        leafref
          (keystore-supported)?
            ++-rw
\+(asymmetric-key-ref)
  \+rw asymmetric-key-ref? leafref
  \ (keystore-supported)? \+rw
  \value?
  \ binary
  \ pported)? \+rw ks:asymmetric key-ref
  \erence? \+rw certificate
  \+rw certificate
    \+rw (local-or-keystore)
      \ (local)
        \+rw local-definition
  \tion
    \+rw algorithm
      \+rw asymmetric-key-algorithm-type
        \+rw public-key
          \+rw binary
            \+rw (private-key)
              \+rw (private-key)
                \+rw (hidden-private-key)
                  \+rw (encrypt-private-key)
                    \+rw (binary)
                      \+rw (empty)
\rypted-private-key
    |        |        |                 |     |        +--rw \
\(key-type)
    |        |        |                 |     |        |  +--\ 
\:(symmetric-key-ref)
    |        |        |                 |     |        |  |  
\+--rw symmetric-key-ref? leafref
    |        |        |                 |     |        |  |  
\     (keystore-supported)?
    |        |        |                 |     |        |     
\:(asymmetric-key-ref)
    |        |        |                 |     |        |     
\+--rw asymmetric-key-ref? leafref
    |        |        |                 |     |        |     
\     (keystore-supported)?
    |        |        |                 |     |        +--rw \
\value?
    |        |        |                 |     |              
\ binary
    |        |        |                 |     |              
\+--rw cert?
    |        |        |                 |     |              
\  end-entity

\ity-cert-cms
    |        |        |                 |     |              
\te-expiration
    |        |        |                 |     |              
\on-date
    |        |        |                 |     |              
\:date-and-time
    |        |        |                 |     |              
\certificate-signing-request
    |        |        |                 |     |              
\ject
    |        |        |                 |     |              
\inary
    |        |        |                 |     |              
\ributes?
    |        |        |                 |     |              
\inary
    |        |        |                 |     |              
\ertificate-signing-request
    |        |        |                 |     |              
\inary
    |        |        |                 |     |              
\pported)?
    |        |        |                 |     |              
\+--:(keystore)
\(keystore-supported)?
\key-ref
\ric-key-ref? leafref
\store-supported?
\key-ref?
\tric-key-ref? leafref
\store-supported?
\me
\signing-request
\ning-request
\ef
\fref

\key-ref
\ric-key-ref? leafref
\store-supported?
\key-ref?
\tric-key-ref? leafref
\store-supported?
\me
\signing-request
\ning-request
\ef
\fref

Watsen                   Expires January 3, 2020               [Page 64]
(local-client-auth-supported)

+--rw ca-certs?
  |  ts:certificates-ref
  |  (ts:x509-certificates)?
+--rw client-certs?
  ts:certificates-ref
  (ts:x509-certificates)?
+--:(external)
  (external-client-auth-supported)

+--rw client-auth-defined-elsewhere?

empty

+--rw cert-maps
  +--rw cert-to-name* [id]
    +--rw id     uint32
    +--rw fingerprint
      x509c2n:tls-fingerprint
    +--rw map-type
      identityref
    +--rw name     string

+--rw hello-params
  (tls-server-hello-params-config)

+--rw tls-versions
  +--rw tls-version* identityref
+--rw cipher-suites
  +--rw cipher-suite* identityref

+--rw keepalives!
  (tls-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
Appendix B. Change Log

B.1. 00 to 01
   o Renamed "keychain" to "keystore".

B.2. 01 to 02
   o Added to ietf-netconf-client ability to connect 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.

B.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).

B.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.

B.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)
B.6.  05 to 06

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

B.7.  06 to 07

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

B.8.  07 to 08

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

B.9.  08 to 09

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

B.10.  09 to 10

- Reformatted YANG modules.
B.11.  10 to 11

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

B.12.  11 to 12

- Removed the "Design Considerations" section.
- Removed the ‘must’ statement limiting keepalives in periodic connections.
- Updated models and examples to reflect removal of the "demux" containers in the imported models.
- Updated the "periodic-connection" description statements to be more like the RESTCONF draft, especially where it described dropping the underlying TCP connection.
- Updated text to better reference where certain examples come from (e.g., which Section in which draft).
- In the server model, commented out the "must ‘pinned-ca-certs or pinned-client-certs’" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.
- Replaced the 'listen', 'initiate', and 'call-home' features with boolean expressions.

B.13.  12 to 13

- Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

B.14.  13 to 14

- Adjusting from change in TLS client model (removing the top-level 'certificate' container), by swapping refining-in a ‘mandatory true’ statement with a ‘must’ statement outside the ‘uses’ statement.
o Updated examples to reflect ietf-crypto-types change (e.g.,
identities --> enumerations)

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
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

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 then 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:

- Supported dampening periods for on-change subscriptions
- Supported (reporting) periods for periodic subscriptions
- Maximum number of objects that can be sent in an update
- 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 cannot 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

- 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.

- in case the capability might change during run-time e.g. due to licensing, HW constraints etc.

- 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

- vendor independent (standard)

- formal (no freeform English text please)

- 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:

- 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.

- 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

  " \n
"
- 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 for 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.";
    }
}
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:

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 following registrations are requested:

name: 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]

[I-D.lengyel-netmod-yang-instance-data]


7.2. Informative References


Appendix A. Changes between revisions

v00 - v01

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

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

- "XXXX" --> the assigned RFC value for this draft
- "AAAA" --> the assigned RFC value for I-D.ietf-netconf-tcp-client-server
- "BBBB" --> the assigned RFC value for I-D.ietf-netconf-tls-client-server
Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

- "2019-07-02" --> the publication date of this draft

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

- 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 January 3, 2020.

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.
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.

YANG feature statements are used to enable implementations to advertise which potentially uncommon 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.

This tree diagram only shows the nodes defined in this module; it does show the nodes defined by "grouping" statements used by this module.

Please see Appendix A.1 for a tree diagram that illustrates what the module looks like with all the "grouping" statements expanded.
module: ietf-restconf-client
  +--rw restconf-client
      +--u restconf-client-grouping

grouping restconf-client-grouping
  +-- initiate! {https-initiate}?
      |  +-- restconf-server* [name]
      |     +-- name? string
      |     +-- endpoints
      |     |  +-- endpoint* [name]
      |     |     +-- name? string
      |     |     +-- (transport)
      |     |     |  +--:(https) {https-initiate}?
      |     |     |     +-- https
      |     |     |     |  +-- tcp-client-parameters
      |     |     |     |      |  +--u tcpc:tcp-client-grouping
      |     |     |     |  +-- tls-client-parameters
      |     |     |     |      |  +--u tlsc:tls-client-grouping
      |     |     |  +-- http-client-parameters
      |     |     |      |  +--u httpc:http-client-grouping
      |     +-- connection-type
      |     |  +-- (connection-type)
      |     |     |  +--:(persistent-connection)
      |     |     |     +-- persistent!
      |     |     |  +--:(periodic-connection)
      |     |     |     +-- periodic!
      |     |     |     |  +-- period? uint16
      |     |     |     |  +-- anchor-time? yang:date-and-time
      |     |     |     |  +-- idle-timeout? uint16
      |     +-- reconnect-strategy
      |     |  +-- start-with? enumeration
      |     |  +-- max-attempts? uint8
  +-- listen! {https-listen}?
      |  +-- idle-timeout? uint16
      |  +-- endpoint* [name]
      |     +-- name? string
      |     +-- (transport)
      |     |  +--:(https) {https-listen}?
      |     |     +-- https
      |     |     |  +-- tcp-server-parameters
      |     |     |      |  +--u tcps:tcp-server-grouping
      |     |     |  +-- tls-client-parameters
      |     |     |      |  +--u tlsc:tls-client-grouping
      |     |     |  +-- http-client-parameters
      |     |     |      |  +--u httpc:http-client-grouping
2.2. Example Usage

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

This example is consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and 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>
            <keepalives>
              <idle-time>15</idle-time>
              <max-probes>3</max-probes>
              <probe-interval>30</probe-interval>
            </keepalives>
            </https>
            </tcp-client-parameters>
            </endpoints>
            </initiate>
            </restconf-server>
            </restconf-client>
</keepalives>
</tls-client-parameters>
</http-client-parameters>
</client-identity>
</http-client-parameters>
</https>
</endpoint>
<endpoint>
<name>corp-fw2.example.com</name>
<https>
<tcp-client-parameters>
<remote-address>corp-fw2.example.com</remote-address>
<keepalives>
<idle-time>15</idle-time>
<max-probes>3</max-probes>
<probe-interval>30</probe-interval>
</keepalives>
</tcp-client-parameters>
<tls-client-parameters>
<classification>
<client-identity>
<local-definition>
<algorithm>rsa2048</algorithm>
<private-key>base64encodedvalue==</private-key>
<public-key>base64encodedvalue==</public-key>
<cert>base64encodedvalue==</cert>
</local-definition>
</client-identity>
<server-authentication>
<ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
<server-certs>explicitly-trusted-server-certs</server-certs>
</server-authentication>
<keepalives>
<max-wait>30</max-wait>
<max-attempts>3</max-attempts>
</keepalives>
</tls-client-parameters>
</http-client-parameters>
</protocol-version>HTTP/1.1</protocol-version>
</client-identity>
</basic>
</client-identity>
</http-client-parameters>
</https>
</endpoint>
<user-id>bob</user-id>
<password>secret</password>
</basic>
</http-client-parameters>
</https>
</endpoint>
</endpoints>
<connection-type>
<persistent/>
</connection-type>
</restconf-server>
</initiate>

<!-- endpoints to listen for RESTCONF Call Home connections on -->
<listen>
  <endpoint>
    <name>Intranet-facing listener</name>
    <https>
      <tcp-server-parameters>
        <local-address>11.22.33.44</local-address>
      </tcp-server-parameters>
      <tls-client-parameters>
        <client-identity>
          <local-definition>
            <algorithm>rsa2048</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
            <cert>base64encodedvalue==</cert>
          </local-definition>
        </client-identity>
        <server-authentication>
          <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
          <server-certs>explicitly-trusted-server-certs</server-certs>
        </server-authentication>
      </tls-client-parameters>
      <http-client-parameters>
        <protocol-version>HTTP/1.1</protocol-version>
        <client-identity>
          <basic>
            <user-id>bob</user-id>
            <password>secret</password>
          </basic>
        </client-identity>
      </http-client-parameters>
    </https>
  </endpoint>
</listen>
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@2019-07-02.yang"
module ietf-restconf-client {
  yang-version 1.1;
  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";
</CODE BEGINS>
This module contains a collection of YANG definitions for configuring RESTCONF 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 (https://trustee.ietf.org/license-info).

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

revision 2019-07-02 {
  description
    "Initial version";
  reference
      "RFC XXXX: RESTCONF Client and Server Models";
}

// Features

feature https-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 https-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 "https-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."
      container tcp-client-parameters {
        description
        "A wrapper around the TCP client parameters to avoid name collisions."
        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."
          }
        }
      }
    }
    container tls-client-parameters {
      must "client-identity" {
        description
        "NETCONF/TLS clients MUST pass some authentication credentials."
      }
    }
    container http-client-parameters {
      description
      "A wrapper around the HTTP client parameters to avoid name collisions."
      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 the
                RESTCONF server, using the reconnection strategy.

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

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

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

                In the case that the previous connection is
still active, establishing a new connection
is NOT RECOMMENDED.";
            }
        }
    }
} // endpoint
} // endpoints
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}:'
    + '(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
} // restconf-server
} // initiate

container listen {
  if-feature "https-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.";
        container tcp-server-parameters {
          description
          "A wrapper around the TCP client parameters to avoid name collisions.";
          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.";
          }
        }
      }
    }
  }
}
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 both
listening for connections as well as initiating call-home
connections.

YANG feature statements are used to enable implementations to
advertise which potentially uncommon parts of the 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.

This tree diagram only shows the nodes defined in this module; it does show the nodes defined by "grouping" statements used by this module.

Please see Appendix A.2 for a tree diagram that illustrates what the module looks like with all the "grouping" statements expanded.

```
module: ietf-restconf-server
  +--rw restconf-server
    +--u restconf-server-app-grouping

      grouping restconf-server-grouping
        +-- client-identification
        +-- cert-maps
          +--u x509c2n:cert-to-name

      grouping restconf-server-listen-stack-grouping
        +-- (transport)
          +--:(http) {http-listen}?
            +-- http
              +-- external-endpoint
                +-- address inet:ip-address
                +-- port? inet:port-number
              +-- tcp-server-parameters
                +--u tcps:tcp-server-grouping
              +-- http-server-parameters
                +--u https:http-server-grouping
              +-- restconf-server-parameters
                +--u rcs:restconf-server-grouping

          +--:(https) {https-listen}?
            +-- https
              +-- tcp-server-parameters
              +--u tcps:tcp-server-grouping
              +-- tls-server-parameters
                +--u tlss:tls-server-grouping
              +-- http-server-parameters
                +--u https:http-server-grouping
              +-- restconf-server-parameters
                +--u rcs:restconf-server-grouping

      grouping restconf-server-callhome-stack-grouping
        +-- (transport)
          +--:(https) {https-listen}?
            +-- https
              +-- tcp-client-parameters
```
3.2. Example Usage

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

This example is consistent with the examples presented in Section 2 of [I-D.ietf-netconf-trust-anchors] and 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>
      <tcp-server-parameters>
        <local-address>11.22.33.44</local-address>
      </tcp-server-parameters>
      <tls-server-parameters>
        <server-identity>
          <local-definition>
            <algorithm>rsa2048</algorithm>
            <private-key>base64encodedvalue==</private-key>
            <public-key>base64encodedvalue==</public-key>
            <cert>base64encodedvalue==</cert>
          </local-definition>
        </server-identity>
        <client-authentication>
          <required/>
          <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
          <client-certs>explicitly-trusted-client-certs</client-certs>
        </client-authentication>
      </tls-server-parameters>
    </https>
  </endpoint>
</listen>

<restconf-server-parameters>
  <client-identification>
    <cert-maps>
      <cert-to-name>
        <id>1</id>
        <fingerprint>11:0A:05:11:00</fingerprint>
        <map-type>x509c2n:san-any</map-type>
      </cert-to-name>
      <cert-to-name>
        <id>2</id>
        <fingerprint>B3:4F:A1:8C:54</fingerprint>
        <map-type>x509c2n:specified</map-type>
        <name>scooby-doo</name>
      </cert-to-name>
    </cert-maps>
  </client-identification>
</restconf-server-parameters>
<call-home>
  <restconf-client>
    <name>config-manager</name>
    <endpoints>
      <endpoint>
        <name>east-data-center</name>
        <https>
          <tcp-client-parameters>
            <remote-address>east.example.com</remote-address>
          </tcp-client-parameters>
          <tls-server-parameters>
            <server-identity>
              <local-definition>
                <algorithm>rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
                <cert>base64encodedvalue==</cert>
              </local-definition>
            </server-identity>
            <client-authentication>
              <required/>
              <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
              <client-certs>explicitly-trusted-client-certs</client-certs>
            </client-authentication>
          </tls-server-parameters>
          <http-server-parameters>
            <server-name>foo.example.com</server-name>
            <protocol-versions>
              <protocol-version>HTTP/1.1</protocol-version>
              <protocol-version>HTTP/2.0</protocol-version>
            </protocol-versions>
          </http-server-parameters>
        </https>
        <restconf-server-parameters>
          <client-identification>
            <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-maps>
          </client-identification>
        </restconf-server-parameters>
      </endpoint>
    </endpoints>
  </restconf-client>
</call-home>
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@2019-07-02.yang"
module ietf-restconf-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-restconf-server";
  prefix rcs;

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

  import ietf-inet-types {
    prefix inet;
    reference
  }

Watsen
Expires January 3, 2020
[Page 23]
"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."
revision 2019-07-02 {
  description
    "Initial version";
  reference
    "RFC XXXX: RESTCONF Client and Server Models";
}

// Features

feature http-listen {
  description
    "The ‘http-listen’ feature indicates that the RESTCONF server supports opening a port to listen for incoming RESTCONF over TPC client connections, whereby the TLS connections are terminated by an external system.";
  reference
    "RFC 8040: RESTCONF Protocol";
}

feature https-listen {
  description
    "The ‘https-listen’ feature indicates that the RESTCONF server supports opening a port to listen for incoming RESTCONF over TLS client connections, whereby the TLS connections are terminated by the server itself/";
  reference
    "RFC 8040: RESTCONF Protocol";
feature https-call-home {
    description
        "The ‘https-call-home’ feature indicates that the RESTCONF
        server supports initiating connections to RESTCONF clients.";
    reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
}

// Groupings

grouping restconf-server-grouping {
    description
        "A reusable grouping for configuring a RESTCONF server
        without any consideration for how underlying transport
        sessions are established.
        
        Note that this grouping uses fairly typical descendent
        node names such that a stack of ‘uses’ statements will
        have name conflicts. It is intended that the consuming
        data model will resolve the issue (e.g., by wrapping
        the ‘uses’ statement in a container called
        ‘restconf-server-parameters’). This model purposely does
        not do this itself so as to provide maximum flexibility
        to consuming models.";

container client-identification {  //FIXME: if-feature?
    description
        "Specifies a mapping through which clients MAY be identified
        (i.e., the RESTCONF username) from a supplied certificate.
        Note that a client MAY alternatively be identified via an
        HTTP-level authentication schema. This configuration does
        not necessitate clients send a certificate (that can be
        controlled via the ietf-restconf-server module).";

container cert-maps {
    uses x509c2n:cert-to-name;
    description
        "The cert-maps container is used by TLS-based RESTCONF
        servers (even if the TLS sessions are terminated
        externally) 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
}
grouping restconf-server-listen-stack-grouping {
  description
  "A reusable grouping for configuring a RESTCONF server
  'listen' protocol stack, for a single connection.";
  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 http {
      if-feature "http-listen";
      container http {
        description
        "Configures RESTCONF server stack assuming that
        TLS-termination is handled externally.";
        container external-endpoint {
          description
          "Identifies contact information for the external
          system that terminates connections before passing
          them thru to this server (e.g., a network address
          translator or a load balancer). These values have
          no effect on the local operation of this server, but
          may be used by the application when needing to
          inform other systems how to contact this server.";
          leaf address {
            type inet:ip-address;
            mandatory true;
            description
            "The IP address or hostname of the external system
            that terminates incoming RESTCONF client
            connections before forwarding them to this
            server.";
          }
          leaf port {
            type inet:port-number;
            default "443";
            description
            "The port number that the external system listens
            on for incoming RESTCONF client connections that
            are forwarded to this server. The default HTTPS
            port (443) is used, as expected for a RESTCONF
            server.";
          }
        }
      }
    }
  }
}
connection.";
}
}
container tcp-server-parameters {
  description
  "A wrapper around the TCP server parameters to avoid name collisions.";
  uses tcps:tcp-server-grouping {
    refine "local-port" {
      default "80";
      description
      "The RESTCONF server will listen on the IANA-assigned well-known port value for 'http' (80) if no value is specified.";
    }
  }
}

container http-server-parameters {
  description
  "A wrapper around the HTTP server parameters to avoid name collisions.";
  uses https:http-server-grouping;
}
container restconf-server-parameters {
  description
  "A wrapper around the RESTCONF server parameters to avoid name collisions.";
  uses rcs:restconf-server-grouping;
}
}
}

case https {
  if-feature "https-listen";
  container https {
    description
    "Configures RESTCONF server stack assuming that TLS-termination is handled internally.";
    container tcp-server-parameters {
      description
      "A wrapper around the TCP server parameters to avoid name collisions.";
      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.";
        }
      }
    }
  }
}
grouping restconf-server-callhome-stack-grouping {
    description
    "A reusable grouping for configuring a RESTCONF server 'call-home' protocol stack, for a single connection.";
    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
                "Configures RESTCONF server stack assuming that TLS-termination is handled internally.";
                container tcp-client-parameters {
                    description
                    "A wrapper around the TCP client parameters to avoid name collisions.";
                    uses tcpc:tcp-client-grouping {
                        refine "remote-port" {
                        }}
                    }
                }
            }
        case tcp {
            container tcp {
                description
                "A wrapper around the TCP client parameters to avoid name collisions.";
                uses tcpc:tcp-client-grouping {
                }
            }
        }
    }
}


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.";
}
}
}
}
}
}
}
}
}
}
}
}
}
}

grouping restconf-server-app-grouping {
description
"A reusable grouping for configuring a RESTCONF server
application that supports both 'listen' and 'call-home'
protocol stacks and for many connections.";
container listen {
  if-feature "https-listen";
presence
  "Enables the RESTCONF server to listen for RESTCONF
client 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.";
}
uses restconf-server-listen-stack-grouping;
}
}
container call-home {
  if-feature "https-call-home";
presence
    "Enables the RESTCONF server to initiate the underlying
     transport connection to RESTCONF clients.";
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.";
}
uses restconf-server-callhome-stack-grouping;
}
}
container connection-type {
  description
    "Indicates the RESTCONF server’s preference for how the
RESTCONF connection is maintained.

choice connection-type {
  mandatory true;
  description
  "Selects between available connection types."
  case persistent-connection {
    container persistent {
      presence "Indicates that a persistent connection is to be maintained.";
      description
      "Maintain a persistent connection to the RESTCONF client. If the connection goes down, immediately start trying to reconnect to the RESTCONF server, using the reconnection strategy.

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

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

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

      In the case that the previous connection is still active (i.e., the RESTCONF client has not closed it yet), establishing a new connection is NOT RECOMMENDED."
    }
  }
  leaf period {
    type uint16;
    units "minutes";
    default "60";
  }
}
description
"Duration of time between periodic connections."
}
leaf anchor-time {
  type yang:date-and-time {
    // constrained to minute-level granularity
    pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}Z'
    + '(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
        "
        "
      }
      enum subsequent {
        description
        "
        "
      }
      enum custom {
        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-app-grouping

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

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

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

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.

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). Some of 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 of the subtrees or data nodes in the modules defined in this
document need to be protected from write operations.

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 of the subtrees or data nodes in the modules defined in this
document need to be protected from read operations.

Some of the RPC operations in the YANG modules 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:
The modules defined in this document do not define any ‘RPC’ or ‘action’ statements.

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:

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

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 following registrations are requested:

name: ietf-restconf-client
prefix: ncc
reference: RFC XXXX

name: ietf-restconf-server
prefix: ncs
reference: RFC XXXX

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore]

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

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


6.2. Informative References

[I-D.ietf-netconf-trust-anchors]


Appendix A. Expanded Tree Diagrams

A.1. Expanded Tree Diagram for 'ietf-restconf-client'

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-restconf-client" module.

This tree diagram shows all the nodes defined in this module, including those defined by "grouping" statements used by this module.

Please see Section 2.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

module: ietf-restconf-client
  +++-rw restconf-client
    +++-rw initiate! {https-initiate}?
      +++-rw restconf-server* [name]
        +++-rw name           string
        +++-rw endpoints
          +++-rw endpoint* [name]
            +++-rw name           string
            +++-rw (transport)
              +++-:(https) {https-initiate}?
                +++-rw https
                  +++-rw tcp-client-parameters
                    +++-rw remote-address    inet:host
                    +++-rw remote-port?      inet:port-number
                    +++-rw local-address?    inet:ip-address
                      (local-binding-supported)?
                    +++-rw local-port?       inet:port-number
                      (local-binding-supported)?
                  +++-rw keepalives!
                    (keepalives-supported)?
                      +++-rw idle-time        uint16
                      +++-rw max-probes       uint16
                      +++-rw probe-interval    uint16
                  +++-rw tls-client-parameters
                    +++-rw client-identity
                      +++-rw (local-or-keystore)
                        +++-:(local)
                          {local-definitions-supported}

rted)?

rithm-t
++-rw keystore-reference
   ++-rw asymmetric-key?
      ks:asymmetric-key-ref

++-rw certificate?
lea

++-rw server-authentication
  ++-rw ca-certs?
     ts:certificates-ref
     (ts:x509-certificates)?
  ++-rw server-certs?
     ts:certificates-ref
     (ts:x509-certificates)?
  ++-rw hello-params
     (tls-client-hello-params-config)

++-rw tls-versions
  ++-rw tls-version* identityref
  ++-rw cipher-suites
     ++-rw cipher-suite* identityref
  ++-rw keepalives!
     (tls-client-keepalives)?
     ++-rw max-wait? uint16
     ++-rw max-attempts? uint8

++-rw http-client-parameters
++-rw protocol-version? enumeration
++-rw client-identity
  ++-rw (auth-type)?
     ++: (basic)
        ++-rw basic {basic-auth}?
           ++-rw user-id? string
           ++-rw password? string
     ++: (bearer)
        ++-rw bearer {bearer-auth}?
           ++-rw token? string
     ++: (digest)
        ++-rw digest {digest-auth}?
           ++-rw username? string
           ++-rw password? string
     ++: (hoba)
        ++-rw hoba {hoba-auth}?
     ++: (mutual)
        ++-rw mutual {mutual-auth}?
     ++: (negotiate)
        ++-rw negotiate {negotiate-auth}?
     ++: (oauth)
        ++-rw oauth {oauth-auth}?
| vate-key)                                                                 | | | | | ---rw encrypted\n-private-key                                                                       | | | | | +---rw (key-t\n-type)                                                                                     | | | | | +--:(symm\n-ometric-key-ref)                                                                     | | | | | | +--rw \n-symmetric-key-ref? leafref  | | | | | +--rw \n (keystore-supported)?                                                               | | | | | +--:(asym\netric-key-ref)                                                                     | | | | | +--rw \n-asymmetric-key-ref? leafref  | | | | | | +--rw \n (keystore-supported)?                                                               | | | | | ---rw value? bina\n-ry                                                                                           | | | | | +---rw cert?                                                                   | | | | | end-entity-cms                                                                          | | | | | +---n certificate-exp\niration                                                             | | | | | +-- expiration-date yang:date-time-\nand-time                                         | | | | | +---x generate-certif\icate-signing-request                                              | | | | | +-----w input                                                                                 | | | | | +-----w subject                                                                                 | | | | | +-----w attribute                                                                                 | | | \s?                                                                                           | | | | | | binary                                                                                           | | | | | +---ro output                                                                                  | | | | | +---ro certificate                                                                                 | | | \te-signing-request                                                                                     | | | | | +--:{keystore-supporte\(keystore-supported)\}                                                  | | | | | +---rw asymmetric-key?                                                                         | | | | | ks:asymmetric\n-key-ref                                                                     | | | | | +---rw certificate?  \n

Watsen                   Expires January 3, 2020               [Page 43]


leafref

```yaml
  +--rw server-authentication
    +--rw ca-certs?
      |   ts:certificates-ref
      |   (ts:x509-certificates)?
    +--rw server-certs?
      ts:certificates-ref
      (ts:x509-certificates)?
  +--rw hello-params
    (tls-client-hello-params-\{

(config)?

```
+++rw max-probes uint16
++rw probe-interval uint16
+++rw tls-client-parameters
+++rw client-identity
   +++rw (local-or-keystore)
      ++:(local)
         {local-definitions-supported}?
         +++rw local-definition
            ++rw algorithm
               | asymmetric-key-algorithm-t
               | binary
            ++rw (private-key-type)
               ++:(private-key)
                  | ++rw public-key
                     | binary
            ++rw (hidden-private-key)
            ++rw hidden-private-key?
               empty
            ++:(encrypted-private-key)
               +++rw encrypted-private-key
                  ++rw (key-type)
                     | ++:(symmetric-key-ref?)
\f)
\y-ref? leafref
   | | | | | | +++rw symmetric-key-ref?
\y-supported)?
   | | | | | | +++:(asymmetric-key-ref?)
\ef)
\ey-ref? leafref
   | | | | | | +++rw asymmetric-key-ref?
\ey-supported)?
   | | | | | | +++rw value?
                  | binary
   | +++rw cert?
                  | end-entity-cert-cms
   | +++n certificate-expiration
      | ++ expiration-date
         | yang:date-and-time
   | +++x generate-certificate-signing-
g-request
   | | | | +++w input
      | | +++w subject binary
      | | +++w attributes? binary
      | +++ro output
         | +++ro certificate-signing-r
```plaintext

---

<table>
<thead>
<tr>
<th>binary</th>
</tr>
</thead>
</table>
| +--:(keystore) {keystore-supported}?
  |   +--rw keystore-reference
  |       +--rw asymmetric-key?
  |          |   ks:asymmetric-key-ref
  |       +--rw certificate?       leafref
| +--rw server-authentication
  |   +--rw ca-certs?
  |       ts:certificates-ref
  |   |   (ts:x509-certificates)?
| +--rw server-certs?
  |   ts:certificates-ref
  |   (ts:x509-certificates)?
| +--rw hello-params
  |   {tls-client-hello-params-config}?
  |   +--rw tls-versions
  |       +--rw tls-version*   identityref
  |       +--rw cipher-suites
  |       +--rw cipher-suite*   identityref
| +--rw keepalives! {tls-client-keepalives}?
  |   +--rw max-wait?       uint16
  |   +--rw max-attempts?   uint8
| +--rw http-client-parameters
| +--rw protocol-version?   enumeration
| +--rw client-identity
|   +--rw (auth-type)?
|     +--:(basic)
|     |   +--rw basic {basic-auth}?
|     |       +--rw user-id?    string
|     |       +--rw password?   string
|     +--:(bearer)
|     |   +--rw bearer {bearer-auth}?
|     |       +--rw token?   string
|     +--:(digest)
|     |   +--rw digest {digest-auth}?
|     |       +--rw username?   string
|     |       +--rw password?   string
|     +--:(hoba)
|     |   +--rw hoba {hoba-auth}?
|     +--:(mutual)
|     |   +--rw mutual {mutual-auth}?
|     +--:(negotiate)
|     |   +--rw negotiate {negotiate-auth}?
|     +--:(oauth)
|     |   +--rw oauth {oauth-auth}?
|     +--:(scram-sha-1)
|     |   +--rw scram-sha-1 {scram-sha-1-auth}?
|     +--:(scram-sha-256)
|   |   +--rw scram-sha-256
```

---

Watsen                   Expires January 3, 2020               [Page 46]
\-key-ref)  
\tric-key-ref?  leafref  
| | | | | +=rw asymme\ 
\store-supported)?  
| | | | | +=rw value?  
| | | | | binary 
| | | | | +=rw cert?  
| | | | | end-entity-cert-cms 
| | | | | -=n certificate-expiration  
| | | | | -= expiration-date  
| | | | | yang:date-and-ti\ 
\me  
\signing-request  
| | | | | +=x generate-certificate-\ 
| | | | | +=w input  
| | | | | -=w subject  
| | | | | binary 
| | | | | -=w attributes?  
| | | | | binary 
| | | | | -=ro output  
| | | | | -=ro certificate-sig\ 
\ning-request  
| | | | | binary 
| | | | | +=x (keystore)  
| | | | | {keystore-supported)?  
| | | | | +=w keystore-reference  
| | | | | +=w asymmetric-key?  
| | | | | ks:asymmetric-key-r\ 
\ef  
\fref  
| | | | | +=x server-authentication  
| | | | | +=x ca-certs?  
| | | | | ts:certificates-ref  
| | | | | {ts:x509-certificates}?  
| | | | | +=x server-certs?  
| | | | | ts:certificates-ref  
| | | | | {ts:x509-certificates}?  
| | | | | +=x hello-params  
| | | | | {tls-client-hello-params-config}\ 
| | | | | +=x tls-versions  
| | | | | |=x tls-version*  identityref  
| | | | | +=x cipher-suites  
| | | | | +=x cipher-suite*  identityref  
| | | | | +=x keepalives!  
| | | | | {tls-client-keepalives}?
A.2. Expanded Tree Diagram for 'ietf-restconf-server'

The following tree diagram [RFC8340] provides an overview of the data model for the "ietf-restconf-server" module.

This tree diagram shows all the nodes defined in this module, including those defined by "grouping" statements used by this module.

Please see Section 3.1 for a tree diagram that illustrates what the module looks like without all the "grouping" statements expanded.

```plaintext
module: ietf-restconf-server
   +++-rw restconf-server
      +++-rw listen! {https-listen}?
         +++-rw endpoint* [name]
            +++-rw name string
            +++-rw (transport)
               +++-:(http) {http-listen}?
                  +++-rw http
                     +++-rw external-endpoint
                        +++-rw address inet:ip-address
                        +++-rw port? inet:port-number
                     +++-rw tcp-server-parameters
                        +++-rw local-address inet:ip-address
                        +++-rw local-port? inet:port-number
                        +++-rw keepalives! {keepalives-supported}?
                           +++-rw idle-time uint16
                           +++-rw max-probes uint16
                           +++-rw probe-interval uint16
                     +++-rw http-server-parameters
                        +++-rw server-name? string
                        +++-rw protocol-versions
                           +++-rw protocol-version* enumeration
                        +++-rw client-authentication!
                           +++-rw (required-or-optional)
                              +++-:(required)
                                 +++-rw required?
                                    empty
                                 +++-:(optional)
                                    +++-rw optional?
```

---

Watsen                   Expires January 3, 2020               [Page 49]
empty
+-rw (local-or-external)
  +--:(local)
    {local-client-auth-supported}?
  +--rw users
    +--rw user* [name]
      +--rw name       string
      +--rw password?  ianach:crypt-hash
  +--:(external)
    {external-client-auth-supported}?

+-rw client-auth-defined-elsewhere?
  empty
+-rw restconf-server-parameters
  +--rw client-identification
    +--rw cert-maps
      +--rw cert-to-name* [id]
        +--rw id          uint32
        +--rw fingerprint
          x509c2n:tls-fingerprint
        +--rw map-type   identityref
        +--rw name       string
  +--:(https) (https-listen)?
    +--rw https
      +--rw tcp-server-parameters
        +--rw local-address  inet:ip-address
        +--rw local-port?    inet:port-number
        +--rw keepalives! {keepalives-supported}?
          +--rw idle-time    uint16
          +--rw max-probes   uint16
          +--rw probe-interval uint16
      +--rw tls-server-parameters
        +--rw server-identity
          +--rw (local-or-keystore)
            +--:(local)
              {local-definitions-supported}?
            +--rw local-definition
              +--rw algorithm
                symmetric-key-algorithm-t
            +--rw public-key
              binary
            +--rw (private-key-type)
              +--:(private-key)
                +--rw private-key?
                  binary
              +--:(hidden-private-key)
                +--rw hidden-private-key?
++-rw ca-certs?
   ts:certificates-ref
   (ts:x509-certificates)?
++-rw client-certs?
   ts:certificates-ref
   (ts:x509-certificates)?
++-:(external)
   {external-client-auth-supported}?
++-rw client-auth-defined-elsewhere?
   empty
++-rw hello-params
   {tls-server-hello-params-config}? 
   +++-rw tls-versions
   |- ++-rw tls-version* identityref
   | ++-rw cipher-suites
   |   +++-rw cipher-suite* identityref
   ++-rw keepalives! (tls-server-keepalives)?
   +++-rw max-wait? uint16
   +++-rw max-attempts? uint8
++-rw http-server-parameters
   +++-rw server-name? string
   +++-rw protocol-versions
   |   +++-rw protocol-version* enumeration
++-rw client-authentication!
   +++-rw (required-or-optional)
   |   +++-:(required)
   |   |   +++-rw required?
   |   |   |   empty
   |   +++-:(optional)
   |   |   +++-rw optional?
   |   |   |   empty
   |   +++-rw (local-or-external)
   |   |   +++-:(local)
   |   |   |   {local-client-auth-supported}?
   |   |   |   +++-rw users
   |   |   |   |   +++-rw user* [name]
   |   |   |   |   |   +++-rw name string
   |   |   |   |   |   +++-rw password?
   |   |   |   |   |   |   ianach:encrypted
   |   |   |   |   |   |   |   {external-client-auth-supported}?
++-rw restconf-server-parameters
   +++-rw client-identification
   +++-rw cert-maps
++-:(optional)
   +++-rw optional?
      empty
   +++-rw (local-or-external)
   +++-:(local)
      {local-client-auth-supported}?
      +++-rw ca-certs?
         ts:certificates-ref
         {ts:x509-certificates}?
      +++-rw client-certs?
         ts:certificates-ref
         {ts:x509-certificates}?
      +++-:(external)
         {external-client-auth-supported}?
      +++-rw client-auth-defined-elsewhere?
         empty
      +++-rw hello-params
         {tls-server-hello-params-config}?
      +++-rw tls-versions
         | +++-rw tls-version* identityref
         | +++-rw cipher-suites
         |   +++-rw cipher-suite* identityref
         +++-rw keepalives!
            {tls-server-keepalives}?
         +++-rw max-wait? uint16
         +++-rw max-attempts? uint8
      +++-rw http-server-parameters
      +++-rw server-name? string
      +++-rw protocol-versions
         | +++-rw protocol-version* enumeration
      +++-rw client-authentication!
      +++-rw (required-or-optional)
         |+++-:(required)
         |   +++-rw required?
         |      empty
         |+++-:(optional)
         |   +++-rw optional?
         |      empty
         +++-rw (local-or-external)
         +++-:(local)
         |   {local-client-auth-supported}?
         +++-rw users
         |   +++-rw user* [name]
---rw name     string
---rw password?  ianach:crypt-hash
          +(external)
            {external-client-auth-supported}?

        +--rw client-auth-defined-else

          empty
        +--rw restconf-server-parameters
          +--rw client-identification
            +--rw cert-maps
              +--rw cert-to-name* [id]
                +--rw id        uint32
                +--rw fingerprint  x509c2n:tls-fingerprint
              +--rw map-type
                |   identityref
                +--rw name     string

        +--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

Appendix B. Change Log

B.1. 00 to 01

  o Renamed "keychain" to "keystore".

B.2. 01 to 02

  o Filled in previously missing 'ietf-restconf-client' module.

  o Updated the ietf-restconf-server module to accommodate new grouping 'ietf-tls-server-grouping'.
B.3.  02 to 03

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

B.4.  03 to 04

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

B.5.  04 to 05

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

B.6.  05 to 06

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

B.7.  06 to 07

- removed "idle-timeout" from "persistent" connection config.
- 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.

B.8. 07 to 08

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

B.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.

B.10. 09 to 10

o Reformatted YANG modules.

B.11. 10 to 11

o Adjusted for the top-level "demux container" added to groupings imported from other modules.

o Added "must" expressions to ensure that keepalives are not configured for "periodic" connections.

o Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

o Moved "expanded" tree diagrams to the Appendix.

B.12. 11 to 12

o Removed the ‘must’ statement limiting keepalives in periodic connections.

o Updated models and examples to reflect removal of the "demux" containers in the imported models.
o Updated the "periodic-connection" description statements to better describe behavior when connections are not closed gracefully.

o Updated text to better reference where certain examples come from (e.g., which Section in which draft).

o In the server model, commented out the "must ‘pinned-ca-certs or pinned-client-certs’" statement to reflect change made in the TLS draft whereby the trust anchors MAY be defined externally.

o Replaced the ’listen’, ’initiate’, and ’call-home’ features with boolean expressions.

B.13. 12 to 13

o Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

o In ietf-restconf-server, Added ‘http-listen’ (not https-listen) choice, to support case when server is behind a TLS-terminator.

o Refactored server module to be more like other ‘server’ models. If folks like it, will also apply to the client model, as well as to both the netconf client/server models. Now the ‘restconf-server-grouping’ is just the RC-specific bits (i.e., the "demux" container minus the container), ’restconf-server-[listen|callhome]-stack-grouping’ is the protocol stack for a single connection, and ’restconf-server-app-grouping’ is effectively what was before (both listen+callhome for many inbound/outbound endpoints).

B.14. 13 to 14

o Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

o Adjusting from change in TLS client model (removing the top-level ’certificate’ container).

o Added "external-endpoint" to the "http-listen" choice in ietf-restconf-server.

Acknowledgements

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

Author’s Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net
YANG Groupings for SSH Clients and SSH Servers

draft-ietf-netconf-ssh-client-server-14

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:

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

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

- "XXXX" -- the assigned RFC value for this draft
- "YYYY" -- the assigned RFC value for I-D.ietf-netconf-trust-anchors
- "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:
The following Appendix section is to be removed prior to publication:

- Appendix A. Change Log

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 ......................................... 9
4. The SSH Server Model .................................... 14
   4.1. Tree Diagram ........................................ 14
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.
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 2 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:

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

<!-- which host-keys will this client trust -->
<server-authentication>
  <ssh-host-keys>explicitly-trusted-ssh-host-keys</ssh-host-keys>
</server-authentication>

<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>
</transport-params>

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

</ssh-client>
The following example configures the client identity using a key from the keystore:
<ssh-client
xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <username>foobar</username>
    <public-key>
      <keystore-reference>ex-rsa-key</keystore-reference>
    </public-key>
  </client-identity>

  <!-- which host-keys will this client trust -->
  <server-authentication>
    <ssh-host-keys>explicitly-trusted-ssh-host-keys</ssh-host-keys>
  </server-authentication>

  <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>
  </transport-params>

  <keepalives>
    <max-wait>30</max-wait>
    <max-attempts>3</max-attempts>
  </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@2019-06-07.yang"
module ietf-ssh-client {
  yang-version 1.1;
  prefix sshc;

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2019-06-07; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
  };

  import ietf-truststore {
    prefix ts;
    reference
      "RFC YYYY: A YANG Data Model for a Truststore";
  };

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

  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: 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.";

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 (https://trustee.ietf.org/license-info).

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

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’, ‘SHALL NOT’, ‘SHOULD’, ‘SHOULD NOT’, ‘RECOMMENDED’, ‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.";

revision 2019-06-07 {
    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.

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

container client-identity {
    nacm:default-deny-write;
    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 {
            nacm:default-deny-all;
            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 "sscmn:ssh-x509-cert";
        uses
        ks:local-or-keystore-end-entity-cert-with-key-grouping;
        description
        "A locally-defined or referenced certificate to be used for client authentication.";
    }
container client-identity

corner container server-authentication {
    must 'ssh-host-keys or ca-certs or server-certs';
    description "Trusted server identities.";
    leaf ssh-host-keys {
        if-feature "ts:ssh-host-keys";
        type ts: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 ca-certs {
    if-feature "sscmn:ssh-x509-certs";
    if-feature "ts:x509-certificates";
    type ts: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 server-certs {
    if-feature "sscmn:ssh-x509-certs";
    if-feature "ts:x509-certificates";
    type ts: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";
}
} // container server-authentication

container transport-params {
  nacm:default-deny-write;
  if-feature "ssh-client-transport-params-config";
  description
    "Configurable parameters of the SSH transport layer.";
  uses sshcmn:transport-params-grouping;
} // container transport-parameters

container keepalives {
  nacm:default-deny-write;
  if-feature "ssh-client-keepalives";
  presence "Indicates that keepalives are enabled.";
  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.";
  }
} // container keepalives
} // grouping ssh-client-grouping

<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

--- server-identity
  +-- host-key* [name]
      +-- name?            string
      +-- (host-key-type)
          +--:(public-key)
              +-- public-key
                  +---u ks:local-or-keystore-asymmetric-key-grouping
                  +--:(certificate)
                      +-- certificate {sshcmn:ssh-x509-certs}?
                          +---u ks:local-or-keystore-end-entity-cert-with-key-grouping

--- client-authentication
  +-- supported-authentication-methods
      +-- publickey?   empty
      +-- passsword?   empty
      +-- hostbased?   empty
      +-- none?        empty
      +-- other*       string
  +--:(local-or-external)
      +--:(local) {local-client-auth-supported}?
          +-- users
              +-- user* [name]
                  +-- name?            string
                  +-- password?         ianach:crypt-hash
                  +-- authorized-key* [name]
                      +-- name?            string
                      +-- algorithm        string
                      +-- key-data         binary
              +--:(external) {external-client-auth-supported}?
                  +-- client-auth-defined-elsewhere?  empty
                  +-- transport-params {ssh-server-transport-params-config}?
                      +--u sshcmn:transport-params-grouping
              +-- 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 2 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) ===========

```xml
<ssh-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
  <!-- which host-keys will this SSH server present -->
  <server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
      </public-key>
    </host-key>
  </server-identity>
  <!-- which client credentials will this SSH server trust -->
  <client-authentication>
    <supported-authentication-methods>
      <publickey/>
    </supported-authentication-methods>
    <!--<local-definition>-->->
    <users>
      <user>
        <name>mary</name>
      </user>
    </users>
    <!--<local-definition>-->->
  </client-authentication>
</ssh-server>
```

<client-certs>explicitly-trusted-client-certs</client-certs>
</client-authentication>

<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>
</transport-params>

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

<ssh-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
  <!-- which host-keys will this SSH server present -->
  <server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <keystore-reference>ex-rsa-key</keystore-reference>
      </public-key>
    </host-key>
  </server-identity>
  <!-- which client credentials will this SSH server trust -->
  <client-authentication>
<supported-authentication-methods>
  <publickey/>
</supported-authentication-methods>
<!--<local-definition>-->
<users>
  <user>
    <name>mary</name>
  </user>
</users>
<!--</local-definition>-->
<!--
<ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
<client-certs>explicitly-trusted-client-certs</client-certs>
--> 
</client-authentication>

<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>
</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@2019-06-07.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 2019-06-07; // stable grouping definitions
        reference
            "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
    }

    import ietf-truststore {
        prefix ta;
        reference
            "RFC YYYY: A YANG Data Model for a Truststore";
    }

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

    import iana-crypt-hash {
        prefix ianach;
        reference
            "RFC 7317: A YANG Data Model for System Management";
    }

    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: 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."
Internet-Draft    Groupings for SSH Clients and Servers        June 2019

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

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

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’,
‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document
are to be interpreted as described in BCP 14 (RFC 2119)
(RFC 8174) when, and only when, they appear in all
capitals, as shown here.

revision 2019-06-07 {
  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.";
}

feature local-client-auth-supported {
  description
    "Indicates that the SSH server supports local configuration
    of client credentials.";
}
feature external-client-auth-supported {
  
  description
  "Indicates that the SSH server supports external configuration
  of client credentials.";
}

// Groupings

grouping ssh-server-grouping {
  
  description
  "A reusable grouping for configuring a SSH server without
  any consideration for how underlying TCP sessions are
  established.

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

  container server-identity {
    
    nacm:default-deny-write;
    
    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";
}
// container server-identity

container client-authentication {
  nacm:default-deny-write;
  description
    "Specifies if SSH client authentication is required or optional, and specifies if the SSH client authentication credentials are configured locally or externally."
  container supported-authentication-methods {
    description
      "Indicates which authentication methods the server supports."
    leaf publickey {
      type empty;
      description
        "Indicates that the ‘publickey’ method is supported. Note that RFC 6187 X.509v3 Certificates for SSH uses the ‘publickey’ method name."
      reference
        RFC 6187: X.509v3 Certificates for Secure Shell Authentication.";
    }
  }
}
leaf passsword {
    type empty;
    description "Indicates that the 'password' method is supported.";
    reference "RFC 4252: The Secure Shell (SSH) Authentication Protocol.";
}
leaf hostbased {
    type empty;
    description "Indicates that the 'hostbased' method is supported.";
    reference "RFC 4252: The Secure Shell (SSH) Authentication Protocol.";
}
leaf none {
    type empty;
    description "Indicates that the 'none' method is supported.";
    reference "RFC 4252: The Secure Shell (SSH) Authentication Protocol.";
}
leaf-list other {
    type string;
    description "Indicates a supported method name not defined by RFC 4253.";
    reference "RFC 4252: The Secure Shell (SSH) Authentication Protocol.";
}
choice local-or-external {
    mandatory true;
    description "Indicates if the client credentials are configured locally or externally.";
    case local {
        if-feature "local-client-auth-supported";
        description "Client credentials are configured locally.";
        container users {
            description "A list of locally configured users.";
            list user {
                key name;
            }
        }
    }
}

The list of local users configured on this device.

leaf name {
  type string;
  description
    "The user name string identifying this entry.";
}

leaf password {
  type ianach:crypt-hash;
  description
    "The password for this entry.";
}

list authorized-key {
  key name;
  description
    "A list of public SSH keys for this user. These keys are allowed for SSH authentication, as described in RFC 4253.";
  reference
    "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
  leaf name {
    type string;
    description
      "An arbitrary name for the SSH key.";
  }
  leaf algorithm {
    type string;
    mandatory true;
    description
      "The public key algorithm name for this SSH key. Valid values are the values in the IANA ‘Secure Shell (SSH) Protocol Parameters’ registry, Public Key Algorithm Names.”;
    reference
      "IANA ‘Secure Shell (SSH) Protocol Parameters’ registry, Public Key Algorithm Names”;
  }
  leaf key-data {
    type binary;
    mandatory true;
    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"
}
}
// list user

if-feature "sshcmn:ssh-x509-certs"
description
"A reference to a list of certificate authority (CA) certificates and a reference to a list of
client certificates."
leaf ca-certs {
  if-feature "ts:x509-certificates"
  type ts:certificates-ref; // local or remote
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 CA certificate."
  reference
  "RFC YYYY: YANG Data Model for Global Trust Anchors"
}
leaf client-certs {
  if-feature "ts:x509-certificates"
  type ts:certificates-ref; // local or remote
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 client certificate."
  reference
  "RFC YYYY: YANG Data Model for Global Trust Anchors"
}

*/

} // container users
} // case local
}

} // case external
}

if-feature "external-client-auth-supported"
} description
"Client credentials are configured externally, such
as via RADIUS, RFC 7317, or another mechanism."
leaf client-auth-defined-elsewhere {
  type empty;
description
"Indicates that client credentials are configured elsewhere."
}
} // choice local-or-external
} // container client-authentication

container transport-params {
  nacm:default-deny-write;
  if-feature "ssh-server-transport-params-config";
  description
  "Configurable parameters of the SSH transport layer.";
  uses sshcmn:transport-params-grouping;
} // container transport-params

container keepalives {
  nacm:default-deny-write;
  if-feature "ssh-server-keepalives";
  presence "Indicates that keepalives are enabled.";
  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.";
  }
} // container keepalives
} // grouping server-identity-grouping
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):
<table>
<thead>
<tr>
<th>sshcmn:host-key-alg</th>
<th>ct:signature-algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsa-shal</td>
<td>dsa-shal</td>
</tr>
<tr>
<td>rsa-pkcs1-shal</td>
<td>rsa-pkcs1-shal</td>
</tr>
<tr>
<td>rsa-pkcs1-sha256</td>
<td>rsa-pkcs1-sha256</td>
</tr>
<tr>
<td>rsa-pkcs1-sha512</td>
<td>rsa-pkcs1-sha512</td>
</tr>
<tr>
<td>ecdsa-secp256r1-sha256</td>
<td>ecdsa-secp256r1-sha256</td>
</tr>
<tr>
<td>ecdsa-secp384r1-sha384</td>
<td>ecdsa-secp384r1-sha384</td>
</tr>
<tr>
<td>ecdsa-secp521r1-sha512</td>
<td>ecdsa-secp521r1-sha512</td>
</tr>
<tr>
<td>x509v3-rsa-pkcs1-shal</td>
<td>x509v3-rsa-pkcs1-shal</td>
</tr>
<tr>
<td>x509v3-rsa2048-pkcs1-sha256</td>
<td>x509v3-rsa2048-pkcs1-sha256</td>
</tr>
<tr>
<td>x509v3-ecdsa-secp256r1-sha256</td>
<td>x509v3-ecdsa-secp256r1-sha256</td>
</tr>
<tr>
<td>x509v3-ecdsa-secp384r1-sha384</td>
<td>x509v3-ecdsa-secp384r1-sha384</td>
</tr>
<tr>
<td>x509v3-ecdsa-secp521r1-sha512</td>
<td>x509v3-ecdsa-secp521r1-sha512</td>
</tr>
</tbody>
</table>

Table 1 The SSH Host-key-alg Compatibility Matrix

<table>
<thead>
<tr>
<th>sshcmn:key-exchange-alg</th>
<th>ct:key-negotiation-algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>diffie-hellman-group14-shal</td>
<td>diffie-hellman-group14-shal</td>
</tr>
<tr>
<td>diffie-hellman-group14-sha256</td>
<td>diffie-hellman-group14-sha256</td>
</tr>
<tr>
<td>diffie-hellman-group15-sha512</td>
<td>diffie-hellman-group15-sha512</td>
</tr>
<tr>
<td>diffie-hellman-group16-sha512</td>
<td>diffie-hellman-group16-sha512</td>
</tr>
<tr>
<td>diffie-hellman-group17-sha512</td>
<td>diffie-hellman-group17-sha512</td>
</tr>
<tr>
<td>diffie-hellman-group18-sha512</td>
<td>diffie-hellman-group18-sha512</td>
</tr>
<tr>
<td>ecdh-sha2-secp256r1</td>
<td>ecdh-sha2-secp256r1</td>
</tr>
<tr>
<td>ecdh-sha2-secp384r1</td>
<td>ecdh-sha2-secp384r1</td>
</tr>
</tbody>
</table>

Table 2 The SSH Key-exchange-alg Compatibility Matrix

<table>
<thead>
<tr>
<th>sshcmn:encryption-alg</th>
<th>ct:symmetric-key-encryption-algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>aes-128-cbc</td>
<td>aes-128-cbc</td>
</tr>
<tr>
<td>aes-192-cbc</td>
<td>aes-192-cbc</td>
</tr>
<tr>
<td>aes-256-cbc</td>
<td>aes-256-cbc</td>
</tr>
<tr>
<td>aes-128-ctr</td>
<td>aes-128-ctr</td>
</tr>
<tr>
<td>aes-192-ctr</td>
<td>aes-192-ctr</td>
</tr>
<tr>
<td>aes-256-ctr</td>
<td>aes-256-ctr</td>
</tr>
</tbody>
</table>

Table 3 The SSH Encryption-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.

```plaintext
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.

```xml
<transport-params
  xmlns="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].

```yml
<CODE BEGINS> file "ietf-ssh-common@2019-06-07.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>"

```
This module defines a common features, identities, and groupings for Secure Shell (SSH).

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 (https://trustee.ietf.org/license-info).

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


revision 2019-06-07 {
  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.";
}
feature ssh-dh-group-exchange {
  description
  "Diffie-Hellman Group Exchange is supported for SSH.";
  reference
}

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.";
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";
}
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 {

Watsen, et al. Expires December 9, 2019 [Page 38]
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 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:

*: All of the nodes defined by the grouping statement in both the "ietf-ssh-client" and "ietf-ssh-server" modules are sensitive to write operations. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., or
even the modification of transport or keepalive parameters can dramatically alter the implemented security policy. For this reason, all the nodes are protected the NACM extension "default-deny-write".

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:

ssh-client-grouping/client-identity/: This subtree in the "ietf-ssh-client" module contains nodes that are additionally sensitive to read operations such that, in normal use cases, they should never be returned to a client. Specifically, the descendent nodes 'password', 'public-key/local-definition/private-key' and 'certificate/local-definition/private-key'. For this reason, all of these node are protected by the NACM extension "default-deny-all".

ssh-server-grouping/server-identity/: This subtree in the "ietf-ssh-server" module contains nodes that are additionally sensitive to read operations such that, in normal use cases, they should never be returned to a client. Specifically, the descendent nodes 'host-key/public-key/local-definition/private-key' and 'host-key/certificate/local-definition/private-key'. For this reason, both of these node are protected by the NACM extension "default-deny-all".

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:

*: The groupings defined in this document include "action" statements that come from groupings defined in [I-D.ietf-netconf-crypto-types]. Please consult that document for the security considerations of the "action" statements defined by the "grouping" statements defined in this document.

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:
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:

<table>
<thead>
<tr>
<th>name:</th>
<th>ietf-ssh-client</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix:</td>
<td>sshc</td>
</tr>
<tr>
<td>reference:</td>
<td>RFC XXXX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name:</th>
<th>ietf-ssh-server</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix:</td>
<td>sshs</td>
</tr>
<tr>
<td>reference:</td>
<td>RFC XXXX</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name:</th>
<th>ietf-ssh-common</th>
</tr>
</thead>
<tbody>
<tr>
<td>prefix:</td>
<td>sshcmn</td>
</tr>
<tr>
<td>reference:</td>
<td>RFC XXXX</td>
</tr>
</tbody>
</table>

8. References

8.1. Normative References

[I-D.ietf-netconf-crypto-types]

[I-D.ietf-netconf-keystore]


8.2. Informative References


BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
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.
Internet-Draft    Groupings for SSH Clients and Servers        June 2019

- Updating descriptions in transport-params-grouping and the
erserver’s usage of it.
- Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- Updated YANG to use typedefs around leafrefs to common keystore
  paths
- Now inlines key and certificates (no longer a leafref to keystore)
A.5. 04 to 05
- Merged changes from co-author.
A.6. 05 to 06
- Updated to use trust anchors from trust-anchors draft (was
  keystore draft)
- Now uses new keystore grouping enabling asymmetric key to be
  either locally defined or a reference to the keystore.
A.7. 06 to 07
- factored the ssh-[client|server]-groupings into more reusable
  groupings.
- 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
- Added a number of compatibility matrices to Section 5 (thanks
  Frank!)
- Clarified that any configured "host-key-alg" values need to be
  compatible with the configured private key.
A.9. 08 to 09
- Updated examples to reflect update to groupings defined in the
  keystore -09 draft.
- Add SSH keepalives features and groupings.
- Prefixed top-level SSH grouping nodes with ‘ssh-‘ and support
  mashups.
Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10
- Reformatted the YANG modules.

A.11. 10 to 11
- Reformatted lines causing folding to occur.

A.12. 11 to 12
- Collapsed all the inner groupings into the top-level grouping.
- Added a top-level "demux container" inside the top-level grouping.
- Added NACM statements and updated the Security Considerations section.
- Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.
- Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

A.13. 12 to 13
- Removed the "demux containers", floating the nacm:default-deny-write to each descendent node, and adding a note to model designers regarding the potential need to add their own demux containers.
- Fixed a couple references (section 2 --> section 3)
- In the server model, replaced <client-cert-auth> with <client-authentication> and introduced 'local-or-external' choice.

A.14. 13 to 14
- Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.///)
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
Abstract

This document defines three YANG modules: the first defines a grouping for configuring a generic TCP client, the second defines a grouping for configuring a generic TCP server, and the third defines a grouping common to the TCP clients and TCP servers.

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:

- "2019-07-02" --> the publication date of this draft

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

- Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."
This Internet-Draft will expire on January 3, 2020.

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 ................................................ 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 ............................................ 7  
4.3. YANG Module .............................................. 8  
5. The TCP Common Model ........................................ 10  
5.1. Tree Diagram ............................................. 10  
5.2. Example Usage ............................................ 11  
5.3. YANG Module .............................................. 11  
6. Security Considerations ..................................... 14  
7. IANA Considerations ......................................... 15  
7.1. The IETF XML Registry .................................. 15  
7.2. The YANG Module Names Registry ......................... 15  
8. References .................................................. 16  
8.1. Normative References ................................... 16  
8.2. Informative References .................................. 16  
Appendix A. Change Log ......................................... 18  
A.1. 00 to 01 ................................................. 18  
A.2. 01 to 02 ................................................. 18  
Authors’ Addresses ............................................. 18
1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines a grouping for configuring a generic TCP client, the second defines a grouping for configuring a generic TCP server, and the third defines a grouping common to the TCP clients and TCP servers.

It is intended that these groupings will be used either standalone, for TCP-based protocols, as part of a stack of protocol-specific configuration models. For instance, these groupings could help define the configuration module for SSH, TLS, or HTTP based applications.

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-binding-supported}?
    +- local-port?       inet:port-number {local-binding-supported}?
    +- keepalives! {keepalives-supported}?
        +- 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.
  <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>
  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-client>

3.3. YANG Module

The ietf-tcp-client YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-client@2019-07-02.yang"
module ietf-tcp-client {
  yang-version 1.1;
  prefix tcpc;

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

  import ietf-tcp-common {
    prefix tcpcmn;
    reference
      "RFC XXXX: YANG Groupings for TCP Clients and TCP Servers";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group and the
     IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

  contact
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    <http://datatracker.ietf.org/wg/tcpm/>
    WG List:  <mailto:netconf@ietf.org>
    <mailto:tcpm@ietf.org>
    Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
              Michael Scharf
              <mailto:michael.scharf@hs-esslingen.de>";

  description

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

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 (https://trustee.ietf.org/license-info).

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


revision 2019-07-02 {
    description
        "Initial version";
    reference
        "RFC XXXX: YANG Groupings for TCP Clients and TCP Servers";
}

// Features

feature local-binding-supported {
    description
        "Indicates that the server supports configuring local bindings (i.e., the local address and local port) for TCP clients.";
}

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.

  Note that this grouping uses fairly typical descendent
  node names such that a stack of ‘uses’ statements will
  have name conflicts. It is intended that the consuming
  data model will resolve the issue (e.g., by wrapping
  the ‘uses’ statement in a container called
  ‘tcp-client-parameters’). This model purposely does
  not do this itself so as to provide maximum flexibility
  to consuming models."
}

leaf remote-address {
  type inet:host;
  mandatory true;
  description
  "The IP address or hostname of the remote peer to
  establish a connection with. 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 establish a
  connection with. An invalid default value (0) is used
  (instead of ‘mandatory true’) so that as application
  level data model may ‘refine’ it with an application
  specific default port number value."
}

leaf local-address {
  if-feature "local-binding-supported";
  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 {
  if-feature "local-binding-supported";
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
    +-- keepalives! (keepalives-supported)?
        +-- 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.
4.3. YANG Module

The ietf-tcp-server YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-server@2019-07-02.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";  
  }  
  import ietf-tcp-common {  
    prefix tcpcmn;  
    reference  
      "RFC XXXX: YANG Groupings for TCP Clients and TCP Servers";  
  }  
  organization  
    "IETF NETCONF (Network Configuration) Working Group and the  
    IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";  
  contact  
    "WG Web:  <http://datatracker.ietf.org/wg/netconf/>  
               <http://datatracker.ietf.org/wg/tcpm/>  
    WG List:  <mailto:netconf@ietf.org>  
              <mailto:tcpm@ietf.org>  
    Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>  
              Michael Scharf  
              <mailto:michael.scharf@hs-esslingen.de>";  
  description  
    "This module defines reusable groupings for TCP servers that  
    can be used as a basis for specific TCP server instances."  
}
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 (https://trustee.ietf.org/license-info).

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


revision 2019-07-02 {
  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."

  Note that this grouping uses fairly typical descendent node names such that a stack of ‘uses’ statements will have name conflicts. It is intended that the consuming data model will resolve the issue (e.g., by wrapping the ‘uses’ statement in a container called

Watsen & Scharf Expires January 3, 2020 [Page 9]
Internet-Draft    Groupings for TCP Clients and Servers        July 2019

'tcp-server-parameters').  This model purposely does not do this itself so as to provide maximum flexibility to consuming models."

leaf local-address {
  type inet:ip-address;
  mandatory true;
  description
  "The local IP address to listen on for incoming TCP 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 an application level data model may 'refine' it with an application specific default port number value."
}

uses tcpcmn:tcp-connection-grouping {
  augment "keepalives" {
    if-feature "tcp-server-keepalives";
    description
    "Add an if-feature statement so that implementations can choose to support TCP server keepalives.";
  }
}

5.  The TCP Common Model

5.1.  Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-tcp-common" module.
module: ietf-tcp-common

grouping tcp-common-grouping
  +++ keepalives! (keepalives-supported)?
  +++ idle-time        uint16
  +++ max-probes       uint16
  +++ probe-interval   uint16

grouping tcp-connection-grouping
  +++ keepalives! (keepalives-supported)?
  +++ idle-time        uint16
  +++ max-probes       uint16
  +++ probe-interval   uint16

5.2. Example Usage

This section presents an example showing the tcp-common-grouping populated with some data.

  <keepalives>
    <idle-time>15</idle-time>
    <max-probes>3</max-probes>
    <probe-interval>30</probe-interval>
  </keepalives>
</tcp-common>

5.3. YANG Module

The ietf-tcp-common YANG module references [RFC6991].

<CODE BEGINS> file "ietf-tcp-common@2019-07-02.yang"
module ietf-tcp-common {
  yang-version 1.1;
  prefix tcpcmn;

  organization
    "IETF NETCONF (Network Configuration) Working Group and the IETF TCP Maintenance and Minor Extensions (TCPM) Working Group";

  contact
    *WG Web:  <http://datatracker.ietf.org/wg/netconf/>
    <http://datatracker.ietf.org/wg/tcpm/>
    WG List:  <mailto:netconf@ietf.org>  
    <mailto:tcpm@ietf.org>
    Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
    Michael Scharf
    <mailto:michael.scharf@hs-esslingen.de>";

This module defines reusable groupings for TCP commons that can be used as a basis for specific TCP common instances.

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 (https://trustee.ietf.org/license-info).

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

revision 2019-07-02 {
    description
        "Initial version";
    reference
        "RFC XXXX: YANG Groupings for TCP Clients and TCP Servers";
}

// Features
feature keepalives-supported {
    description
        "Indicates that keepalives are supported.";
}

// Groupings
grouping tcp-common-grouping {
    description
        "A reusable grouping for configuring TCP parameters common to TCP connections as well as the operating system as a whole.";
    container keepalives {
        if-feature "keepalives-supported";
        presence "Indicates that keepalives are enabled.";
description
"Configures the keep-alive policy, to proactively test the
aliveness of the TCP peer. An unresponsive TCP peer is
dropped after approximately (idle-time * 60) + (max-probes
* probe-interval) seconds."
leaf idle-time {
  type uint16 {
    range "1..max";
  }
  units "seconds";
  mandatory true;
  description
  "Sets the amount of time after which if no data has been
  received from the TCP peer, a TCP-level probe message
  will be sent to test the aliveness of the TCP peer.";
}
leaf max-probes {
  type uint16 {
    range "1..max";
  }
  mandatory true;
  description
  "Sets the maximum number of sequential keep-alive probes
  that can fail to obtain a response from the TCP peer
  before assuming the TCP peer is no longer alive.";
}
leaf probe-interval {
  type uint16 {
    range "1..max";
  }
  units "seconds";
  mandatory true;
  description
  "Sets the time interval between failed probes.";
}
} // container keepalives
} // grouping tcp-common-grouping

grouping tcp-connection-grouping {
  description
  "A reusable grouping for configuring TCP parameters common
to TCP connections.";
  uses tcp-common-grouping;
}

/*
The following is for a future bis...
This comment is here now so as support discussion with TCPM. This comment will be removed before publication.

Should future system-level parameters be defined as a grouping or a container?

grouping tcp-system-grouping {
  description
      "A reusable grouping for configuring TCP parameters common to the operating system as a whole."

  // currently just a placeholder

} /*

} <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, TCP) 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 of the writable/creatable/deletable data nodes in the YANG modules defined in this document are considered more sensitive or vulnerable then standard configuration.

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 of the readable data nodes in the YANG modules defined in this document are considered more sensitive or vulnerable than standard configuration.

This document does not define any RPC actions and hence this section does not consider the security of RPCs.

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:

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

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 following registrations are requested:

name: ietf-tcp-common
prefix: tcpcmn
reference: RFC XXXX

name: ietf-tcp-client
prefix: tcpc
reference: RFC XXXX

name: ietf-tcp-server
prefix: tcps
reference: RFC XXXX
8. References

8.1. Normative References


8.2. Informative References


Watsen & Scharf Expires January 3, 2020 [Page 16]
Appendix A. Change Log

A.1. 00 to 01

- Added 'local-binding-supported' feature to TCP-client model.
- Added 'keepalives-supported' feature to TCP-common model.
- Added 'external-endpoint-values' container and 'external-endpoints' feature to TCP-server model.

A.2. 01 to 02

- Removed the 'external-endpoint-values' container and 'external-endpoints' feature from the TCP-server model.

Authors’ Addresses

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

Michael Scharf
Hochschule Esslingen - University of Applied Sciences
EMail: michael.scharf@hs-esslingen.de
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:

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

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

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

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
A 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.

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
  +-- client-identity
  |   +---- u ks:local-or-keystore-end-entity-cert-with-key-grouping
  +-- server-authentication
    |   +-- ca-certs? ts:certificates-ref
    |       {ts:x509-certificates}?
    |   +-- server-certs? ts:certificates-ref
    |       {ts:x509-certificates}?
    +-- hello-params {tls-client-hello-params-config}?
    |   +-- u tlscmn:hello-params-grouping
    +-- 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 2 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:


<!-- how this client will authenticate itself to the server -->
<client-identity>
  <local-definition>
    <algorithm>rsa2048</algorithm>
    <public-key>base64encodedvalue==</public-key>
    <private-key>base64encodedvalue==</private-key>
    <cert>base64encodedvalue==</cert>
  </local-definition>
</client-identity>

<!-- which certificates will this client trust -->
<server-authentication>
  <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
  <server-certs>explicitly-trusted-server-certs</server-certs>
</server-authentication>

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

</tls-client>

The following example configures the client identity using a key from the keystore:
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@2019-07-02.yang"
module ietf-tls-client {
  yang-version 1.1;
  prefix tlsc;

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2019-07-02; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-truststore {
    prefix ts;
    reference
      "RFC YYYY: A YANG Data Model for a Truststore";
  }

  import ietf-keystore {

prefix ks;
reference
"RFC ZZZZ: A YANG Data Model for a Keystore";
}

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

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

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 (https://trustee.ietf.org/license-info).

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here."

revision 2019-07-02 {

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."

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

  container client-identity { // FIXME: what about PSKs?
    nacm:default-deny-write;
    description
      "A locally-defined or referenced end-entity certificate, including any configured intermediate certificates, the TLS client will present when establishing a TLS connection in its Certificate message, as defined in Section 7.4.2 in RFC 5246.";
    reference
  }
}
uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
} // container client-identity

container server-authentication {  // FIXME: what about PSKs?
  nacm:default-deny-write;
  must 'ca-certs or server-certs';
  description
    "Trusted server identities.";
  leaf ca-certs {
    if-feature "ts:x509-certificates";
    type ts: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 CA certificate.";
  }
  leaf server-certs {
    if-feature "ts:x509-certificates";
    type ts: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 server certificate.";
  }
} // container server-authentication

container hello-params {
  nacm:default-deny-write;
  if-feature "tls-client-hello-params-config";
  uses tlscmn:hello-params-grouping;
  description
    "Configurable parameters for the TLS hello message.";
} // container hello-params

container keepalives {
  nacm:default-deny-write;
  if-feature "tls-client-keepalives";
  presence "Indicates that keepalives are enabled.";
  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 {

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
  +- server-identity
     |  +--- u ks:local-or-keystore-end-entity-cert-with-key-grouping
  +- client-authentication!
     |  +- (required-or-optional)
        |     +--:(required)
        |     |   +-- required?        empty
        |     +--:(optional)
        |         +-- optional?      empty
     |  +- (local-or-external)
        |     +--:(local) {local-client-auth-supported}?
        |           |   +-- ca-certs?  ts:certificates-ref
        |           |   |      {ts:x509-certificates}?
        |           |   +-- client-certs?  ts:certificates-ref
        |           |      {ts:x509-certificates}?
        |     +--:(external) {external-client-auth-supported}?
        |           +-- client-auth-defined-elsewhere?  empty
  +- hello-params {tls-server-hello-params-config}?
     |  +--- u tlscmn:hello-params-grouping
     +- 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 2 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:
<tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">
  <!-- how this server will authenticate itself to the client -->
  <server-identity>
    <local-definition>
      <algorithm>rsa2048</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
      <cert>base64encodedvalue==</cert>
    </local-definition>
  </server-identity>

  <!-- which certificates will this server trust -->
  <client-authentication>
    <required/>
    <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
    <client-certs>explicitly-trusted-client-certs</client-certs>
  </client-authentication>
</tls-server>

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

<tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">
  <!-- how this server will authenticate itself to the client -->
  <server-identity>
    <keystore-reference>
      <asymmetric-key>ex-rsa-key</asymmetric-key>
      <certificate>ex-rsa-cert</certificate>
    </keystore-reference>
  </server-identity>

  <!-- which certificates will this server trust -->
  <client-authentication>
    <required/>
    <ca-certs>explicitly-trusted-client-ca-certs</ca-certs>
    <client-certs>explicitly-trusted-client-certs</client-certs>
  </client-authentication>
</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].
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 (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself for full legal notices.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

revision 2019-07-02 {
  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.";
}

feature local-client-auth-supported {
  description
    "Indicates that the TLS server supports local configuration of client credentials.";
}

feature external-client-auth-supported {
  description
    "Indicates that the TLS server supports external
configuration of client credentials.

// Groupings

grouping tls-server-grouping {
  description
  "A reusable grouping for configuring a TLS server without
  any consideration for how underlying TCP sessions are
  established.

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

  container server-identity {  // FIXME: what about PSKs?
    nacm:default-deny-write;
    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;
  } // container server-identity

  container client-authentication {  // FIXME: what about PSKs?
    nacm:default-deny-write;
    presence
    "Indicates that certificate based client authentication
    is supported (i.e., the server will request that the
    client send a certificate)."
    description
    "Specifies if TLS client authentication is required or
    optional, and specifies if the certificates needed to
authenticate the TLS client are configured locally or externally. If configured locally, the data model enables both trust anchors and end-entity certificate to be set.

choice required-or-optional {
  mandatory true; // or default to ‘required’?
  description "Indicates if TLS-level client authentication is required or optional. This is necessary for some protocols (e.g., RESTCONF) the may optionally authenticate a client via TLS-level authentication, HTTP-level authentication, or both simultaneously)."
  leaf required {
    type empty;
    description "Indicates that TLS-level client authentication is required."
  }
  leaf optional {
    type empty;
    description "Indicates that TLS-level client authentication is optional."
  }
}

case local-or-external {
  mandatory true;
  description "Indicates if the certificates needed to authenticate the client are configured locally or externally. The need to support external configuration for client authentication stems from the desire to support consuming data models that prefer to place client authentication with client definitions, rather than in a data model principally concerned with configuring the transport."
  choice local-or-remote {
    if-feature "local-client-auth-supported";
    description "The certificates needed to authenticate the clients are configured locally."
    leaf ca-certs {
      if-feature "ts:x509-certificates";
      type ts:certificates-ref;//FIXME: local-or-remote?
      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 CA certificate.

reference
"RFC YYYY: YANG Data Model for Global Trust Anchors";

} // leaf client-certs
if-feature "ts:x509-certificates";
type ts:certificates-ref;//FIXME: local-or-remote?
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 client certificate."

reference
"RFC YYYY: YANG Data Model for Global Trust Anchors";

} // case external { if-feature "external-client-auth-supported";
description
"The certificates needed to authenticate the clients are configured externally.";
leaf client-auth-defined-elsewhere {
type empty;
description
"Indicates that certificates needed to authenticate clients are configured elsewhere.";
}
} // choice local-or-external
} // container client-authentication

container hello-params {
nacm:default-deny-write;
if-feature "tls-server-hello-params-config";
uses tlscmn:hello-params-grouping;
description
"Configurable parameters for the TLS hello message.";
} // container hello-params

container keepalives {
nacm:default-deny-write;
if-feature "tls-server-keepalives";
presence "Indicates that keepalives are enabled.";
description
"Configures the keep-alive policy, to proactively test the aliveness of the TLS client. An unresponsive
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.
Table 1-1 TLS 1.2 Compatibility Matrix Part 1: cipher-suites mapping to hash-algorithm

<table>
<thead>
<tr>
<th>cipher-suites in hello-params-grouping</th>
<th>HASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>sha-384</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>sha-384</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</td>
<td>sha-384</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>sha-384</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_CCM</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_CCM</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_CCM</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_CCM</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>sha-384</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>Cipher-suites in hello-params-grouping</td>
<td>Symmetric</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>enc-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>enc-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>enc-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>enc-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</td>
<td>enc-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</td>
<td>enc-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>enc-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>enc-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_CCM</td>
<td>enc-aes-128-ccm</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_CCM</td>
<td>enc-aes-256-ccm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_CCM</td>
<td>enc-aes-128-ccm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_CCM</td>
<td>enc-aes-256-ccm</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>enc-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>enc-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>enc-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>enc-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>enc-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>enc-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>enc-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256</td>
<td>enc-aes-128-gcm</td>
</tr>
</tbody>
</table>

Table 1-2 TLS 1.2 Compatibility Matrix Part 2: cipher-suites mapping to symmetric-key-encryption-algorithm
### Table 1-3 TLS 1.2 Compatibility Matrix Part 3: ciper-suites mapping to MAC-algorithm

<table>
<thead>
<tr>
<th>ciper-suites in hello-params-grouping</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>mac-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_GCM_SHA256</td>
<td>mac-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>mac-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>mac-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</td>
<td>mac-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</td>
<td>mac-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>mac-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>mac-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_CCM</td>
<td>mac-aes-128-ccm</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_CCM</td>
<td>mac-aes-256-ccm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_CCM</td>
<td>mac-aes-128-ccm</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_CCM</td>
<td>mac-aes-256-ccm</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>mac-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>mac-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>mac-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>mac-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>mac-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>mac-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>mac-aes-256-gcm</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256</td>
<td>mac-aes-128-ccm</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_256_CCM_SHA384</td>
<td>mac-aes-256-ccm</td>
</tr>
<tr>
<td>cipher-suites in hello-params-grouping</td>
<td>signature</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>rsa-pkcs1-sha256</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>rsa-pkcs1-sha384</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</td>
<td>ecdsa-secp256r1-sha256</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</td>
<td>ecdsa-secp384r1-sha384</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>rsa-pkcs1-sha256</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>rsa-pkcs1-sha384</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_CCM</td>
<td>rsa-pkcs1-sha256</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_CCM</td>
<td>rsa-pkcs1-sha384</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_CCM</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_CCM</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>rsa-pkcs1-sha256</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>ecdsa-secp256r1-sha256</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>rsa-pkcs1-sha256</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>rsa-pkcs1-sha384</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>N/A</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_CCM</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 1-4 TLS 1.2 Compatibility Matrix Part 4: cipher-suites mapping to signature-algorithm
### Table 1-5 TLS 1.2 Compatibility Matrix Part 5: cipher-suites mapping to key-negotiation-algorithm

<table>
<thead>
<tr>
<th>cipher-suites in hello-params-grouping</th>
<th>key-negotiation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>psk-dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>psk-dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</td>
<td>ecdhe-secp256r1, ...</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</td>
<td>ecdhe-secp256r1, ...</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256</td>
<td>ecdhe-secp256r1, ...</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</td>
<td>ecdhe-secp256r1, ...</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_128_CCM</td>
<td>dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_AES_256_CCM</td>
<td>dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_128_CCM</td>
<td>psk-dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_AES_256_CCM</td>
<td>psk-dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>ecdhe-secp256r1, ...</td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>ecdhe-secp256r1, ...</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256</td>
<td>dhe-ffdhe2048, ...</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>psk-ecdhe-secp256r1,...</td>
</tr>
<tr>
<td>TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256</td>
<td>psk-ecdhe-secp256r1,...</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</td>
<td>psk-ecdhe-secp256r1,...</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384</td>
<td>psk-ecdhe-secp256r1,...</td>
</tr>
<tr>
<td>TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256</td>
<td>psk-ecdhe-secp256r1,...</td>
</tr>
</tbody>
</table>

### Table 2-1 TLS 1.3 Compatibility Matrix Part 1: cipher-suites mapping to hash-algorithm

<table>
<thead>
<tr>
<th>cipher-suites in hello-params-grouping</th>
<th>HASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_AES_128_GCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_AES_256_GCM_SHA384</td>
<td>sha-384</td>
</tr>
<tr>
<td>TLS_CHACHA20_POLY1305_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>TLS_AES_128_CCM_SHA256</td>
<td>sha-256</td>
</tr>
<tr>
<td>cipher-suites in hello</td>
<td>symmetric</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>TLS_AES_128_GCM_SHA256</td>
<td>enc-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_AES_256_GCM_SHA384</td>
<td>enc-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_CHACHA20_POLY1305_SHA256</td>
<td>enc-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_AES_128_CCM_SHA256</td>
<td>enc-aes-128-ccm</td>
</tr>
</tbody>
</table>

Table 2-2 TLS 1.3 Compatibility Matrix Part 2: cipher-suites mapping to symmetric-key-encryption-algorithm

<table>
<thead>
<tr>
<th>cipher-suites in hello</th>
<th>symmetric</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_AES_128_GCM_SHA256</td>
<td>mac-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_AES_256_GCM_SHA384</td>
<td>mac-aes-128-gcm</td>
</tr>
<tr>
<td>TLS_CHACHA20_POLY1305_SHA256</td>
<td>mac-chacha20-poly1305</td>
</tr>
<tr>
<td>TLS_AES_128_CCM_SHA256</td>
<td>mac-aes-128-ccm</td>
</tr>
</tbody>
</table>

Table 2-3 TLS 1.3 Compatibility Matrix Part 3: cipher-suites mapping to MAC-algorithm

<table>
<thead>
<tr>
<th>signatureScheme in hello</th>
<th>signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsa-pkcs1-sha256</td>
<td>rsa-pkcs1-sha256</td>
</tr>
<tr>
<td>rsa-pkcs1-sha384</td>
<td>rsa-pkcs1-sha384</td>
</tr>
<tr>
<td>rsa-pkcs1-sha512</td>
<td>rsa-pkcs1-sha512</td>
</tr>
<tr>
<td>rsa-pss-rsae-sha256</td>
<td>rsa-pss-rsae-sha256</td>
</tr>
<tr>
<td>rsa-pss-rsae-sha384</td>
<td>rsa-pss-rsae-sha384</td>
</tr>
<tr>
<td>rsa-pss-rsae-sha512</td>
<td>rsa-pss-rsae-sha512</td>
</tr>
<tr>
<td>rsa-pss-pss-sha256</td>
<td>rsa-pss-pss-sha256</td>
</tr>
<tr>
<td>rsa-pss-pss-sha384</td>
<td>rsa-pss-pss-sha384</td>
</tr>
<tr>
<td>rsa-pss-pss-sha512</td>
<td>rsa-pss-pss-sha512</td>
</tr>
<tr>
<td>ecdsa-secp256r1-sha256</td>
<td>ecdsa-secp256r1-sha256</td>
</tr>
<tr>
<td>ecdsa-secp384r1-sha384</td>
<td>ecdsa-secp384r1-sha384</td>
</tr>
<tr>
<td>ecdsa-secp521r1-sha512</td>
<td>ecdsa-secp521r1-sha512</td>
</tr>
<tr>
<td>ed25519</td>
<td>ed25519</td>
</tr>
<tr>
<td>ed448</td>
<td>ed448</td>
</tr>
</tbody>
</table>

Table 2-4 TLS 1.3 Compatibility Matrix Part 4: SignatureScheme mapping to signature-algorithm
<table>
<thead>
<tr>
<th>supported Groups in hello</th>
<th>key-negotiation</th>
</tr>
</thead>
<tbody>
<tr>
<td>dhe-ffdhe2048</td>
<td>dhe-ffdhe2048</td>
</tr>
<tr>
<td>dhe-ffdhe3072</td>
<td>dhe-ffdhe3072</td>
</tr>
<tr>
<td>dhe-ffdhe4096</td>
<td>dhe-ffdhe4096</td>
</tr>
<tr>
<td>dhe-ffdhe6144</td>
<td>dhe-ffdhe6144</td>
</tr>
<tr>
<td>dhe-ffdhe8192</td>
<td>dhe-ffdhe8192</td>
</tr>
<tr>
<td>psk-dhe-ffdhe2048</td>
<td>psk-dhe-ffdhe2048</td>
</tr>
<tr>
<td>psk-dhe-ffdhe3072</td>
<td>psk-dhe-ffdhe3072</td>
</tr>
<tr>
<td>psk-dhe-ffdhe4096</td>
<td>psk-dhe-ffdhe4096</td>
</tr>
<tr>
<td>psk-dhe-ffdhe6144</td>
<td>psk-dhe-ffdhe6144</td>
</tr>
<tr>
<td>psk-dhe-ffdhe8192</td>
<td>psk-dhe-ffdhe8192</td>
</tr>
<tr>
<td>ecdhe-secp256r1</td>
<td>ecdhe-secp256r1</td>
</tr>
<tr>
<td>ecdhe-secp384r1</td>
<td>ecdhe-secp384r1</td>
</tr>
<tr>
<td>ecdhe-secp521r1</td>
<td>ecdhe-secp521r1</td>
</tr>
<tr>
<td>ecdhe-x25519</td>
<td>ecdhe-x25519</td>
</tr>
<tr>
<td>ecdhe-x448</td>
<td>ecdhe-x448</td>
</tr>
<tr>
<td>psk-ecdhe-secp256r1</td>
<td>psk-ecdhe-secp256r1</td>
</tr>
<tr>
<td>psk-ecdhe-secp384r1</td>
<td>psk-ecdhe-secp384r1</td>
</tr>
<tr>
<td>psk-ecdhe-secp521r1</td>
<td>psk-ecdhe-secp521r1</td>
</tr>
<tr>
<td>psk-ecdhe-x25519</td>
<td>psk-ecdhe-x25519</td>
</tr>
<tr>
<td>psk-ecdhe-x448</td>
<td>psk-ecdhe-x448</td>
</tr>
</tbody>
</table>

Table 2-5 TLS 1.3 Compatibility Matrix Part 5: Supported Groups mapping to key-negotiation-algorithm

Note that in Table 1-5:

- dhe-ffdhe2048, ... is the abbreviation of dhe-ffdhe2048, dhe-ffdhe3072, dhe-ffdhe4096, dhe-ffdhe6144, dhe-ffdhe8192;
- psk-dhe-ffdhe2048, ... is the abbreviation of psk-dhe-ffdhe2048, psk-dhe-ffdhe3072, psk-dhe-ffdhe4096, psk-dhe-ffdhe6144, psk-dhe-ffdhe8192;
- ecdhe-secp256r1, ... is the abbreviation of ecdhe-secp256r1, ecdhe-secp384r1, ecdhe-secp521r1, ecdhe-x25519, ecdhe-x448;
- 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"
  <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:rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>tlscmn:rsa-with-3des-ede-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@2019-07-02.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";

description
"This module defines a common features, identities, and
groupings for Transport Layer Security (TLS).

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

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

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’,
‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document
are to be interpreted as described in BCP 14 (RFC 2119)
(RFC 8174) when, and only when, they appear in all
capitals, as shown here.”;

revision 2019-07-02 {
  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
}

feature tls-1_2 {
description
"TLS Protocol Version 1.2 is supported."
reference
}

feature tls-1_3 {
description
"TLS Protocol Version 1.2 is supported."
reference
}

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
}

feature tls-3des {
description
"The Triple-DES block cipher is supported for TLS.";
reference
}

feature tls-gcm {

Internet-Draft  Groupings for TLS Clients and Servers  July 2019

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
 Version 1.1";
}

identity tls-1.2 {
 base tls-version-base;
 if-feature "tls-1_2";
 description
 "TLS Protocol Version 1.2.";
 reference
}
}

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
}

identity rsa-with-aes-256-cbc-sha {
  base cipher-suite-base;
  description
  "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA."
  reference
}

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
}

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
}

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
}

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
}

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
}

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
}

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
}

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
  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";
}
}

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 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 subtree defined by the grouping statement in both the "ietf-ssh-client" and "ietf-ssh-server" modules is sensitive to write operations. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., or even the modification of transport or keepalive parameters can dramatically alter the implemented security policy. For this reason, this node is protected the NACM extension "default-deny-write".
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:

/tls-client-parameters/client-identity/: This subtree in the "ietf-tls-client" module contains nodes that are additionally sensitive to read operations such that, in normal use cases, they should never be returned to a client. Some of these nodes (i.e., public-key/local-definition/private-key and certificate/local-definition/private-key) are already protected by the NACM extension "default-deny-all" set in the "grouping" statements defined in [I-D.ietf-netconf-crypto-types].

/tls-server-parameters/server-identity/: This subtree in the "ietf-tls-server" module contains nodes that are additionally sensitive to read operations such that, in normal use cases, they should never be returned to a client. All of these nodes (i.e., host-key/public-key/local-definition/private-key and host-key/certificate/local-definition/private-key) are already protected by the NACM extension "default-deny-all" set in the "grouping" statements defined in [I-D.ietf-netconf-crypto-types].

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:

*: The groupings defined in this document include "action" statements that come from groupings defined in [I-D.ietf-netconf-crypto-types]. Please consult that document for the security considerations of the "action" statements defined by the "grouping" statements defined in this document.

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:
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
prefix:     tlsc
reference:  RFC XXXX

name:       ietf-tls-server
prefix:     tlss
reference:  RFC XXXX

name:       ietf-tls-common
prefix:     tlscmn
reference:  RFC XXXX

8. References

8.1. Normative References

[I-D.ietf-netconf-crypto-types]

[I-D.ietf-netconf-keystore]
[I-D.ietf-netconf-trust-anchors]


8.2. Informative References


BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
Appendix A. Change Log

A.1. 00 to 01

- Noted that ‘0.0.0.0’ and ‘::’ might have special meanings.
- Renamed "keychain" to "keystore".

A.2. 01 to 02

- Removed the groupings containing transport-level configuration. Now modules contain only the transport-independent groupings.
- Filled in previously incomplete ’ietf-tls-client’ module.
- Added cipher suites for various algorithms into new ’ietf-tls-common’ module.

A.3. 02 to 03

- Added a ‘must’ statement to container ‘server-auth’ asserting that at least one of the various auth mechanisms must be specified.
- Fixed description statement for leaf ‘trusted-ca-certs’.

A.4. 03 to 04

- Updated title to "YANG Groupings for TLS Clients and TLS Servers"
- Updated leafref paths to point to new keystore path
- Changed the YANG prefix for ietf-tls-common from ‘tlscom’ to ‘tlscmn’.
- Added TLS protocol versions 1.0 and 1.1.
- Made author lists consistent
- Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- Updated YANG to use typedefs around leafrefs to common keystore paths
- Now inlines key and certificates (no longer a leafref to keystore)
A.5. 04 to 05
- Merged changes from co-author.

A.6. 05 to 06
- Updated to use trust anchors from trust-anchors draft (was keystore draft)
- Now Uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

A.7. 06 to 07
- factored the tls-[client|server]-groupings into more reusable groupings.
- added if-feature statements for the new "x509-certificates" feature defined in draft-ietf-netconf-trust-anchors.

A.8. 07 to 08
- Added a number of compatibility matrices to Section 5 (thanks Frank!)
- Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

A.9. 08 to 09
- Updated examples to reflect update to groupings defined in the keystore draft.
- Add TLS keepalives features and groupings.
- Prefixed top-level TLS grouping nodes with ‘tls-’ and support mashups.
- Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.10. 09 to 10
- Reformatted the YANG modules.
A.11.  10 to 11

- Collapsed all the inner groupings into the top-level grouping.
- Added a top-level "demux container" inside the top-level grouping.
- Added NACM statements and updated the Security Considerations section.
- Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.
- Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

A.12.  11 to 12

- In server model, made 'client-authentication' a 'presence' node indicating that the server supports client authentication.
- In the server model, added a 'required-or-optional' choice to 'client-authentication' to better support protocols such as RESTCONF.
- In the server model, added a 'local-or-external' choice to 'client-authentication' to better support consuming data models that prefer to keep client auth with client definitions than in a model principally concerned with the "transport".
- In both models, removed the "demux containers", floating the nacm:default-deny-write to each descendent node, and adding a note to model designers regarding the potential need to add their own demux containers.
- Fixed a couple references (section 2 --> section 3)

A.13.  12 to 13

- Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.///)

A.14.  12 to 13

- Removed 'container' under 'client-identity' to match server model.
- Updated examples to reflect change grouping in keystore module.
A.15. 13 to 14

- Removed the "certificate" container from "client-identity" in the ietf-tls-client module.

- Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

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
A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-05

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:

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

- "2019-06-07" --> the publication date of this draft

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

- Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute...
working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 9, 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 .................................... 14
4. IANA Considerations ....................................... 14
   4.1. The IETF XML Registry .................................. 14
   4.2. The YANG Module Names Registry ......................... 15
5. References ................................................ 15
   5.1. Normative References ................................... 15
   5.2. Informative References ................................ 15
Appendix A. Change Log ....................................... 17
   A.1. 00 to 01 .............................................. 17
   A.2. 01 to 02 .............................................. 17
   A.3. 02 to 03 .............................................. 17
   A.4. 03 to 04 .............................................. 17
   A.5. 04 to 05 .............................................. 17
Acknowledgements ............................................. 18
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-truststore" module.

```
module: ietf-truststore
  +--rw truststore
    +--rw certificates* [name] {x509-certificates}?
      |   +--rw name          string
      |   +--rw description?  string
      |   +--rw certificate* [name]
      |       |   +--rw name          string
      |       |       +--rw cert          trust-anchor-cert-cms
      |       |       +---n certificate-expiration
      |       |       |     +-- expiration-date     yang:date-and-time
      |       |   +--rw host-keys* [name] {ssh-host-keys}?
      |       |       +--rw name          string
```
++-rw description? string
++-rw host-key* [name]
    +++-rw name string
    +++-rw host-key ct:ssh-host-key

grouping local-or-truststore-certs-grouping
    +-- (local-or-truststore)
      +--:(local) [local-definitions-supported]?
        +-- local-definition
          +-- cert* trust-anchor-cert-cms
            +---n certificate-expiration
              ++ expiration-date yang:date-and-time
          +--:(truststore) {truststore-supported,x509-certificates}?
            ++ truststore-reference? ts:certificates-ref

grouping local-or-truststore-host-keys-grouping
    +-- (local-or-truststore)
      +--:(local) [local-definitions-supported]?
        +-- local-definition
          +-- host-key* ct:ssh-host-key
          +-- cert* trust-anchor-cert-cms
            +---n certificate-expiration
              ++ expiration-date yang:date-and-time
          +--:(truststore) {truststore-supported,ssh-host-keys}?
            ++ truststore-reference? ts:host-keys-ref

grouping truststore-grouping
    +-- certificates* [name] {x509-certificates}?
        +-- name? string
        +-- description? string
        +-- certificate* [name]
            +-- name? string
            +-- cert trust-anchor-cert-cms
                +---n certificate-expiration
                    ++ expiration-date yang:date-and-time
        +-- host-keys* [name] {ssh-host-keys}?
            +-- name? string
            +-- description? string
            +-- host-key* [name]
                +-- name? string
                +-- 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.
<truststore
xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore"
xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">

<!-- Manufacturer's trusted root CA certs -->
<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>
  <certificate>
    <name>Manufacturer Root CA cert 1</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
  <certificate>
    <name>Manufacturer Root CA cert 2</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
</certificates>

<!-- specific end-entity certs for authenticating servers -->
<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 CA.
  </description>
  <certificate>
    <name>Fred Flintstone</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
</certificates>

<!-- trusted CA certs for authenticating servers -->
<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>
  <certificate>
    <name>ca.example.com</name>
  </certificate>
</certificates>
<cert>base64encodedvalue==</cert>
</certificate>
</certificates>

<!-- specific end-entity certs for authenticating clients -->
<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 CA.
  </description>
  <certificate>
    <name>George Jetson</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
</certificates>

<!-- trusted CA certs for authenticating clients -->
<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>
  <certificate>
    <name>ca.example.com</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
</certificates>

<!-- trusted CA certs for random HTTPS servers on Internet -->
<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>
  <certificate>
    <name>ex-certificate-authority</name>
    <cert>base64encodedvalue==</cert>
  </certificate>
</certificates>

<!-- specific SSH host keys for authenticating clients -->
<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>
  <host-key>
    <name>corp-fw1</name>
    <host-key>base64encodedvalue==</host-key>
  </host-key>
</host-keys>

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>
  <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
    <certificates>
      <name>explicitly-trusted-client-certs</name>
      <certificate>
        <name>George Jetson</name>
        <certificate-expiration>
          <expiration-date>2018-08-05T14:18:53-05:00</expiration-date>
        </certificate-expiration>
      </certificate>
    </certificates>
  </truststore>
</notification>

2.3. YANG Module

This YANG module imports modules from [RFC8341] and [I-D.ietf-netconf-crypto-types].

<CODE BEGINS> file "ietf-truststire@2019-06-07.yang"
module ietf-truststore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-truststore";
  prefix ts;
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 truststore to centralize management
    of trust anchors including both X.509 certificates and
    SSH host keys.

    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

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

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL','
    'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
    'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
    are to be interpreted as described in BCP 14 (RFC 2119)
    (RFC 8174) when, and only when, they appear in all
    capitals, as shown here."

revision 2019-06-07 {
    description
feature truststore-supported {
  description
    "The 'truststore-supported' feature indicates that the
    server supports the Truststore.";
}

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

feature x509-certificates {
  description
    "The 'x509-certificates' feature indicates that the server
    implements the /truststore/certificates subtree.";
}

feature ssh-host-keys {
  description
    "The 'ssh-host-keys' feature indicates that the server
    implements the /truststore/host-keys subtree.";
}

typedef certificates-ref {
  type leafref {
    path "/ts:truststore/ts:certificates/ts:name";
  }
  description
    "This typedef enables modules to easily define a reference
to a set of certificates defined in the Truststore.";
}

typedef host-keys-ref {
  type leafref {

path "/ts:truststore/ts:host-keys/ts:name";
}

description
"This typedef enables modules to easily define a reference
to a set of host keys defined in the truststore.";
}

/******************
/*   Groupings   */
/******************

grouping local-or-truststore-certs-grouping {

description
"A grouping that expands to allow trust anchors to be
either stored locally, within the using data model, or be
a reference to trust anchors stored in the truststore.";

choice local-or-truststore {
  mandatory true;
  case local {
    if-feature "local-definitions-supported";
    container local-definition {
      description
      "Container to hold the local trust anchor definitions. A list is defined so as to be symmetric with the truststore definition.";
      uses ct:trust-anchor-certs-grouping;
    }
  }
  case truststore {
    if-feature "truststore-supported";
    if-feature "x509-certificates";
    leaf truststore-reference {
      type ts:certificates-ref;
      description
      "A reference to a set of trust anchors that exists in the truststore.";
    }
  }
}

description
"A choice between an inlined definition and a definition that exists in the truststore.";
}

}


grouping local-or-truststore-host-keys-grouping {

description
"A grouping that expands to allow trust anchors to be
either stored locally, within the using data model, or be
a reference to trust anchors stored in the truststore."
choice local-or-truststore {
  mandatory true;
  case local {
    if-feature "local-definitions-supported";
    container local-definition {
      description
        "Container to hold the local trust anchor definitions.
         A list is defined so as to be symmetric with the
         truststore definition."
      leaf-list host-key {
        nacm:default-deny-write;
        type ct:ssh-host-key;
        description
          "The binary data for this host key."
        reference
          "RFC YYYY: Common YANG Data Types for Cryptography"
      }
      uses ct:trust-anchor-certs-grouping;
    }
  }
  case truststore {
    if-feature "truststore-supported";
    if-feature "ssh-host-keys";
    leaf truststore-reference {
      type ts:host-keys-ref;
      description
        "A reference to a set of trust anchors that exists
         in the truststore."
    }
  }
  description
    "A choice between an inlined definition and a definition
     that exists in the truststore."
}

grouping truststore-grouping {
  description
    "Grouping definition enables use in other contexts. If ever
done, implementations SHOULD augment new 'case' statements
into local-or-keystore 'choice' statements to supply leafrefs
to the new location."
  list certificates {
    if-feature "x509-certificates";
    key "name";
    description

"A list of certificates. These certificates can be used by a server to authenticate clients, or by a client to authenticate servers. Each list of 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 certificates for when authenticating RESTCONF client connections."

leaf name {
  type string;
  description
    "An arbitrary name for this list of certificates.";
}

leaf description {
  type string;
  description
    "An arbitrary description for this list of certificates.";
}

list certificate {
  key "name";
  description
    "A certificate.";
  leaf name {
    type string;
    description
      "An arbitrary name for this certificate. The name must be unique across all lists of 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 host-keys {
  if-feature "ssh-host-keys";
  key "name";
  description
    "A list of host keys. These host-keys can be used by clients to authenticate SSH servers. Each list of 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 host keys for when authentication fails using the default list of host keys."
}
authenticating specific SSH servers.

leaf name {
    type string;
    description
        "An arbitrary name for this list of SSH
         host keys."
}

leaf description {
    type string;
    description
        "An arbitrary description for this list of SSH
         host keys."
}

list host-key {
    key "name";
    description
        "A host key."
    leaf name {
        type string;
        description
            "An arbitrary name for this host-key. Must be
             unique across all lists of 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 host key."
        reference
            "RFC YYYY: Common YANG Data Types for Cryptography"
    }
}

/****************************/
/* Protocol accessible nodes */
/****************************/

container truststore {
    nacm:default-deny-write;
    description
        "The truststore contains sets of X.509 certificates and
         SSH host keys."
    uses truststore-grouping;
}
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:

```
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 following registration is requested:

name:         ietf-truststore
prefix:       ta
reference:    RFC XXXX

5. References

5.1. Normative References


5.2. Informative References


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.

A.4.  03 to 04
  o Added groupings 'local-or-truststore-certs-grouping' and 'local-or-truststore-host-keys-grouping', matching similar definitions in the keystore draft. Note new (and incomplete) "truststore" usage!
  o Related to above, also added features 'truststore-supported' and 'local-trust-anchors-supported'.

A.5.  04 to 05
  o Renamed "trust-anchors" to "truststore"
  o Removed "pinned." prefix everywhere, to match truststore rename
  o Moved everything under a top-level 'grouping' to enable use in other contexts.
  o Renamed feature from 'local-trust-anchors-supported' to 'local-definitions-supported' (same name used in keystore)
  o Removed the "require-instance false" statement from the "*_ref" typedefs.
  o Added missing "ssh-host-keys" and "x509-certificates" if-feature statements
Acknowledgements

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

Author’s Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net
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:

- "2019-06-07" --> the publication date of this draft

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

- Appendix A. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 9, 2019.
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.
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
   +-- protocol-version?   enumeration
   +-- client-identity
      |  +-- (auth-type)?
      |     +-- (basic)
      |     |  +-- user-id?   string
      |     |  +-- password?   string
      |     +-- (bearer)
      |     |  +-- bearer (bearer-auth)?
      |     |     +-- token?   string
      |     +-- (digest)
      |     |  +-- digest (digest-auth)?
      |     |     +-- username?   string
      |     |     +-- password?   string
      |     +-- (hoba)
      |     |  +-- hoba (hoba-auth)?
      |     +-- (mutual)
      |     |  +-- mutual (mutual-auth)?
      |     +-- (negotiate)
      |     |  +-- negotiate (negotiate-auth)?
      |     +-- (oauth)
      |     |  +-- oauth (oauth-auth)?
      |     |     +-- username?   string
      |     +-- (scram-sha-1)
      |     |  +-- scram-sha-1 (scram-sha-1-auth)?
      |     +-- (scram-sha-256)
      |     |  +-- scram-sha-256 (scram-sha-256-auth)?
      +-- (vapid)
          +-- vapid (vapid-auth)?
    +-- proxy-server! {proxy-connect}?
        +-- tcp-client-parameters
        |  +-- u tcp:tcp-client-grouping
        +-- tls-client-parameters
        |  +-- u tls:tls-client-grouping
        +-- proxy-client-identity
        |  +-- user-id?   string
        |  +-- password?   string

3.2. Example Usage

This section presents an example showing the http-client-grouping populated with some data.
3.3. YANG Module

This YANG module has normative references to [RFC6991].

```
  <protocol-version>HTTP/1.1</protocol-version>
  <client-identity>
    <basic>
      <user-id>bob</user-id>
      <password>secret</password>
    </basic>
  </client-identity>
</http-client>
```
can be used as a basis for specific HTTP client instances.

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

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

The key words 'MUST', ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’,
‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document
are to be interpreted as described in BCP 14 (RFC 2119)
(RFC 8174) when, and only when, they appear in all
capitals, as shown here.

revision 2019-06-07 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for HTTP Clients and HTTP Servers";
}

// Features

feature proxy-connect {
  description
    "Proxy connection configuration is configurable for
     HTTP clients on the server implementing this feature.";
}

feature basic-auth {
  description
    "fixme";
}

feature bearer-auth {
  description
    "fixme";
}
feature digest-auth {
  description "fixme";
}

feature hoba-auth {
  description "fixme";
}

feature mutual-auth {
  description "fixme";
}

feature negotiate-auth {
  description "fixme";
}

feature oauth-auth {
  description "fixme";
}

feature scram-sha-1-auth {
  description "fixme";
}

feature scram-sha-256-auth {
  description "fixme";
}

feature vapid-auth {
  description "fixme";
}

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

leaf protocol-version {
  nacm:default-deny-write;
  type enumeration {
    enum HTTP/1.0 {
      description "The client should use the 'HTTP/1.0' protocol.";
    }
    enum HTTP/1.1 {
      description "The client should use the 'HTTP/1.1' protocol.";
    }
    enum HTTP/2.0 {
      description "The client should use the 'HTTP/2.0' protocol.";
    }
  }
  description "The HTTP protocol version the client should use.";
} // leaf protocol-version

container client-identity {
  nacm:default-deny-write;
  description "The credentials used by the client to authenticate to the HTTP server.";
  choice auth-type {
    description "The authentication type.";
    container basic {
      if-feature "basic-auth";
      leaf user-id {
        type string;
        description "The user-id for the authenticating client.";
      }
      leaf password {
        nacm:default-deny-all;
        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 {
  if-feature "bearer-auth";
  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 {
  if-feature "digest-auth";
  leaf username {
    type string;
    description
    "The username for the authenticating client."
  }
  leaf password {
    nacm:default-deny-all;
    type string;
    description
    "The password for the authenticating client."
  }
  description
  "The 'digest' HTTP scheme credentials."
  reference
  "RFC 7616: HTTP Digest Access Authentication"
}
container hoba {
  if-feature "hoba-auth";
  // FIXME
  description
  "The 'hoba' HTTP scheme credentials."
  reference
  "RFC 7486: HTTP Origin-Bound Authentication (HOBA)"
container mutual {
  if-feature "mutual-auth";
  // FIXME
  description
  "The ‘mutual’ HTTP scheme credentials.";
  reference
  "RFC 8120: Mutual Authentication Protocol for HTTP";
}

container negotiate {
  if-feature "negotiate-auth";
  // FIXME
  description
  "The ‘negotiate’ HTTP scheme credentials.";
  reference
  "RFC 4559: SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows";
}

container oauth {
  if-feature "oauth-auth";
  // FIXME
  description
  "The ‘oauth’ HTTP scheme credentials.";
  reference
  "RFC 6749: The OAuth 2.0 Authorization Framework";
}

container scram-sha-1 {
  if-feature "scram-sha-1-auth";
  // FIXME
  description
  "The ‘scram-sha-1’ HTTP scheme credentials.";
  reference
  "RFC 7804: Salted Challenge Response HTTP Authentication Mechanism";
}

container scram-sha-256 {
  if-feature "scram-sha-256-auth";
  // FIXME
  description
  "The ‘scram-sha-256’ HTTP scheme credentials.";
  reference
  "RFC 7804: Salted Challenge Response HTTP Authentication Mechanism";
}

container vapid {
  if-feature "vapid-auth";
  // FIXME
  description
  "The ‘vapid’ HTTP scheme credentials.";
reference
"RFC 8292: Voluntary Application Server Identification (VAPID) for Web Push";
}
}
} // container client-identity

container proxy-server {
nacm:default-deny-write;
if-feature "proxy-connect";
presence true; // only so ex-http-client can pass validation?
container tcp-client-parameters {
    description
    "A wrapper around the TCP parameters to avoid name collisions.";
    uses "tcpc:tcp-client-grouping";
}
container tls-client-parameters {
    description
    "A wrapper around the TLS parameters to avoid name collisions.";
    uses "tlsc:tls-client-grouping";
}
container proxy-client-identity {
    leaf user-id {
        type string;
        description
        "The user-id for the authenticating client.";
    }
    leaf password {
        nacm:default-deny-all;
        type string;
        description
        "The password for the authenticating client.";
    }
    description
    "The 'basic' HTTP scheme credentials.";
    reference
    "RFC 7617: The 'Basic' HTTP Authentication Scheme";
}

description
"Proxy server settings.";
} // container proxy-server
} // grouping http-client-grouping

<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
  += server-name?       string
  += protocol-versions
    |= += protocol-version*   enumeration
    |= += client-authentication!
      |  += (required-or-optional)
      |    +=:(required)       empty
      |    +=:(optional)       empty
      |  += (local-or-external)
      |    +=:(local) {local-client-auth-supported}?
      |         += users
      |         |    += user* [name]
      |         |      += name?   string
      |         |      += password?  ianach:crypt-hash
      |  +=:(external) {external-client-auth-supported}?
      |    += client-auth-defined-elsewhere?   empty

4.2. Example Usage

This section presents an example showing the http-server-grouping populated with some data.

  <server-name>foo.example.com</server-name>
  <protocol-versions>
    <protocol-version>HTTP/1.1</protocol-version>
    <protocol-version>HTTP/2.0</protocol-version>
  </protocol-versions>
</http-server>

4.3. YANG Module

This YANG module has normative references to [RFC6991].

<CODE BEGINS> file "ietf-http-server@2019-06-07.yang"
module ietf-http-server {
  yang-version 1.1;
}
prefix https;

import iana-crypt-hash {
    prefix ianach;
    reference
        "RFC 7317: A YANG Data Model for System Management";
}

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 reusable groupings for HTTP servers that
    can be used as a basis for specific HTTP server instances.

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

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

The key words ’MUST’, ’MUST NOT’, ’REQUIRED’, ’SHALL’,
’SHALL NOT’, ’SHOULD’, ’SHOULD NOT’, ’RECOMMENDED’,
’NOT RECOMMENDED’, ’MAY’, and ’OPTIONAL’ in this document
are to be interpreted as described in BCP 14 (RFC 2119)
(RFC 8174) when, and only when, they appear in all
capitals, as shown here.";
revision 2019-06-07 {
  description
    "Initial version";
  reference
    "RFC XXXX: YANG Groupings for HTTP Clients and HTTP Servers";
}

// Features

feature local-client-auth-supported {
  description
    "Indicates that the HTTP server supports local configuration of client credentials.";
}

feature external-client-auth-supported {
  description
    "Indicates that the HTTP server supports external configuration of client credentials.";
}

// Groupings

grouping http-server-grouping {
  description
    "A reusable grouping for configuring an HTTP server.

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

  leaf server-name {
    nacm:default-deny-write;
    type string;
    description
      "The value of the 'Server' header field. If not set, then underlying software’s default value is used. Set to the empty string to disable.";
  }

  container protocol-versions {
    nacm:default-deny-write;
  }
}
description
"A list of HTTP protocol versions supported by this server."
leaf-list protocol-version {
type enumeration {
enum "HTTP/1.0" {
  description
  "The server supports the 'HTTP/1.0' protocol."
}
enum "HTTP/1.1" {
  description
  "The server supports the 'HTTP/1.1' protocol."
}
enum "HTTP/2.0" {
  description
  "The server supports the 'HTTP/2.0' protocol."
}
}

description
"An HTTP protocol version supported by this server."
}
}

container client-authentication {
nacm:default-deny-write;
presence
"Indicates that HTTP based client authentication is supported (i.e., the server will request that the HTTP client send authenticate when needed). This is needed as some HTTP-based protocols may only support, e.g., TLS-level client authentication."

description
"Specifies if HTTP client authentication is required or optional, and specifies if the credentials needed to authenticate the HTTP client are configured locally or externally."
choice required-or-optional {
  mandatory true; // or default to 'required' ?
description
  "Indicates if HTTP-level client authentication is required or optional. This is necessary for some protocols (e.g., RESTCONF) that may optionally authenticate a client via TLS-level authentication, HTTP-level authentication, or both simultaneously."
leaf required {
  type empty;
description
  "Indicates that HTTP-level client authentication is
required to access protected resources.
}
leaf optional {
  type empty;
  description
  "Indicates that HTTP-level client authentication is optional to access protected resources."
}
}
choice local-or-external {
  mandatory true;
  description
  "Indicates if the client credentials are configured locally or externally. The need to support external configuration for client authentication stems from the desire to support consuming data models that prefer to place client authentication with client definitions, rather than in a data model principally concerned with configuring the transport."
  case local {
    if-feature "local-client-auth-supported";
    description
    "Client credentials are configured locally.";
    container users {
      description
      "A list of locally configured users."
      list user {
        key name;
        description
        "The list of local users configured on this device."
      }
      leaf name {
        type string;
        description
        "The user name string identifying this entry."
      }
      leaf password {
        type ianaCh:encrypt-hash;
        description
        "The password for this entry."
      }
    }
  }
  case external {
    if-feature "external-client-auth-supported";
    description
    "Client credentials are configured externally.";
  }
}
leaf client-auth-defined-elsewhere {
  type empty;
  description
    "Indicates that credentials needed to authenticate
     clients are configured elsewhere."
}
} // choice local-or-external
} // container client-authentication

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:

    FIXME: (pending - TBD)

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:

    FIXME: (pending client auth params?)
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:

The modules defined in this document do not define any 'RPC' or 'action' statements.

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:

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

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
prefix:       httpc
reference:    RFC XXXX

name:         ietf-http-server
prefix:       https
reference:    RFC XXXX

7. References

7.1. Normative References

7.2. Informative References


Author’s Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net
An HTTPS-based Transport for Configured Subscriptions
draft-mahesh-netconf-https-notif-00

Abstract

This document defines a YANG data module for configuring HTTPS based configured subscription, as defined in I-D.ietf-netconf-subscribed-notifications. The use of HTTPS maximizes transport-level interoperability, while allowing for encoding selection from text, e.g. XML or JSON, to binary.

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 December 28, 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
Introduction

Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications] defines a YANG data module for configuring subscribed notifications. It is even defined a subscriptions container that contains a list of receivers. But it defers the configuration and management of those receivers to other documents. This document defines a YANG [RFC7950] data module for configuring and managing HTTPS based receivers for the notifications. Such a configured receiver can be a third party collector, collecting events on behalf of receivers that want to correlate events from different publishers. Configured subscriptions enable a server, acting as a publisher of notifications, to proactively push notifications to external receivers without the receivers needing to first connect to the server, as is the case with dynamic subscriptions.

This document describes how to enable the transmission of YANG modeled notifications, in the configured encoding (i.e., XML, JSON) over HTTPS. The use of HTTPS maximizes transport-level interoperability, while the encoding selection pivots between implementation simplicity (XML, JSON) and throughput (text versus binary).
1.1. Note to RFC Editor

This document uses several placeholder values throughout the document. Please replace them as follows and remove this section before publication.

RFC XXXX, where XXXX is the number assigned to this document at the time of publication.

2019-06-26 with the actual date of the publication of this document.

1.2. Abbreviations

+-----------------+----------------------------------+
| Acronym | Expansion                        |
+-----------------+----------------------------------+
| HTTP      | Hyper Text Transport Protocol    |
| TCP       | Transmission Control Protocol    |
| TLS       | Transport Layer Security         |
+-----------------+----------------------------------+

1.3. 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 [RFC2119] RFC8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3.1. Subscribed Notifications

The following terms are defined in Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications].

- Subscribed Notifications

2. YANG module

2.1. Overview

The YANG module is a definition of a set of receivers that are interested in the notifications published by the publisher. The module contains the TCP, TLS and HTTPS parameters that are needed to communicate with the receiver. The module augments the Subscribed Notifications [I-D.ietf-netconf-subscribed-notifications] receiver
container to create a reference to a receiver defined by the YANG module.

An abridged tree diagram representing the module is shown below.

```
module: ietf-https-notif
  +--rw receivers
    +--rw receiver* [name]
      +--rw name           string
      +--rw tcp-params
        |   +--rw remote-address    inet:host
        |   +--rw remote-port?     inet:port-number
        |   +--rw local-address?   inet:ip-address
        |   +--rw local-port?      inet:port-number
        |   +--rw keepalives!
        |     ...
      +--rw tls-params
        |   +--rw client-identity
        |     ...
        |   +--rw server-authentication
        |     ...
        |   +--rw hello-params {tls-client-hello-params-config}?
        |     ...
        |   +--rw keepalives! {tls-client-keepalives}?
        |     ...
      +--rw http-params
        +--rw protocol-version?   enumeration
        +--rw client-identity
        |   ...
        +--rw proxy-server! {proxy-connect}?
        |   ...

augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
  +--rw receiver-ref?   -> /receivers/receiver/name
```

2.2. YANG module

The YANG module is shown below.

```
<CODE BEGINS> file "ietf-https-notif@2019-06-26.yang"
module ietf-https-notif {
    yang-version 1.1;
    prefix "hsn";

    import ietf-subscribed-notifications {
        prefix sn;
```

import ietf-tcp-client {
    prefix tcpc;
}

import ietf-tls-client {
    prefix tlsc;
}

import ietf-http-client {
    prefix httpc;
}

organization
    "IETF NETCONF Working Group";

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

    Authors: Mahesh Jethanandani (mjethanandani at gmail dot com)
    Kent Watsen (kent plus ietf at watsen dot net)";

description
    "YANG module for configuring HTTPS base configuration.

    Copyright (c) 2018 IETF Trust and the persons identified as
    the document authors. 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 "2019-06-26" {
    description
        "Initial Version.";
    reference
        "RFC XXXX, YANG Data Module for HTTPS Notifications.";
}

identity https {

base sn:transport;

description
  "HTTPS transport for notifications.";

} container receivers {
  list receiver {
    key "name";

    leaf name {
      type string;
      description
        "";
    }

  container tcp-params {
    uses tcpc:tcp-client-grouping;
    description
      "TCP client parameters.";
  }

  container tls-params {
    uses tlsc:tls-client-grouping;
    description
      "TLS client parameters.";
  }

  container http-params {
    uses httpc:http-client-grouping;
    description
      "HTTP client parameters.";
  }

  description
    "All receivers interested in this notification.";

  description
    "HTTPS based notifications.";

} augment "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
  leaf receiver-ref {
    type leafref {
      path "/receivers/receiver/name";
    }

    description
      "Reference to a receiver.";
  }

  description
    "";

}
"Augment the subscriptions container to define the receiver."
;

3. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via 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 [RFC8446]. The NETCONF Access Control Model (NACM) [RFC8341] 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.

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:

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:

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:

4. IANA Considerations

This document registers one URI and one YANG module.

4.1. URI Registration

in the IETF XML registry [RFC3688] [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

4.2. YANG Module Name Registration

This document registers three YANG module in the YANG Module Names registry YANG [RFC6020].

name: ietf-https-notif
prefix: hn
reference: RFC XXXX

5. Examples

This section tries to show some examples in how the model can be used.

5.1. HTTPS Configured Subscription

This example shows how a HTTPS client can be configured to send notifications to a receiver at address 192.0.2.1, port 443 with server certificates, and the corresponding trust store that is used to authenticate a connection.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <receivers
    <receiver>
      <name>foo</name>
      <tcp-params>
        <remote-address>192.0.2.1</remote-address>
        <remote-port>443</remote-port>
        <local-address>192.0.3.1</local-address>
        <local-port>63001</local-port>
      </tcp-params>
      <tls-params>
        <server-authentication>
          <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
          <server-certs>explicitly-trusted-server-certs</server-certs>
        </server-authentication>
      </tls-params>
    </receiver>
  </receivers>
```


<subscriptions
  xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notification">

  <subscription>
    <id>6666</id>
    <stream-subtree-filter>foo</stream-subtree-filter>
    <stream>some-stream</stream>
    <receivers>
      <receiver>
        <name>my-receiver</name>
        <receiver-ref
      </receiver>
    </receivers>
  </subscription>

</subscriptions>

<truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
  <certificates>
    <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>
    <certificate>
      <name>Fred Flintstone</name>
      <cert>base64encodedvalue==</cert>
    </certificate>
  </certificates>

  <certificates>
    <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>
    <certificate>
      <name>ca.example.com</name>
      <cert>base64encodedvalue==</cert>
    </certificate>
  </certificates>
</truststore>
</config>
6. Contributors

7. Acknowledgements

8. Normative references

[I-D.ietf-netconf-subscribed-notifications]


Authors' Addresses

Mahesh Jethanandani
VMware

Email: mjethanandani@gmail.com

Kent Watsen
Watsen Networks
USA

Email: kent+ietf@watsen.net
Abstract

This document describes the distributed data export 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 January 8, 2020.
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.
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.
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,

- the Subscriber generates the subscription instructions to express what and how the collector want to receive the data;
- 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:

- 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).

- Contracts are needed between the Master and each Agent on the Component Capability, and the format for streaming data structure.

- The Master relays the component subscriptions to the Agents.

- 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:

- Component subscriptions are system-controlled, i.e. managed by the Master, not by the subscriber.
- Component subscription settings such as time periods, dampening periods, encodings, receivers adopt the settings of their Global Subscription.
- 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.
- 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 MUST be sent to indicate the new list of Publishers.

8. YANG Tree
module: ietf-multiple-stream-originators
augment /sn:subscriptions/sn:subscription:
  +--ro message-generator-id*   string
  +--ro (transport-access) ?
    +--: (restconf-access)
    +--ro uri*   inet:uri
augment /sn:subscription-started:
  +--ro message-generator-id*   string
augment /sn:subscription-modified:
  +--ro message-generator-id*   string
augment /sn:establish-subscription/sn:output:
  +--ro message-generator-id*   string
  +--ro (transport-access) ?
    +--: (restconf-access)
    +--ro uri*   inet:uri
augment /sn:modify-subscription/sn:output:
  +--ro message-generator-id*   string
  +--ro (transport-access) ?
    +--: (restconf-access)
    +--ro uri*   inet:uri

9.  YANG Module

<CODE BEGINS> file "ietf-multiple-stream-originators@2019-07-07.yang"
module ietf-multiple-stream-originators {
  yang-version 1.1;
  namespace
  prefix mso;
  import ietf-subscribed-notifications {
    prefix sn;
  }

  import ietf-inet-types {
    prefix inet;
  }

  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

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

revision 2019-07-07 {
  description
    "Initial version";
  reference
    "RFC XXXX: Subscription to Multiple Stream Originators";
}

grouping message-generator-ids {
  description
    "Provides a reusable list of message-generator-ids.";

  leaf-list message-generator-id {
    type string;
    config false;
    ordered-by user;
    description
      "Software entity which created the message (e.g.,
       linecard 1). This field is used to notify the
       collector the working originator.";
  }
}

grouping resource-access-list {
  description
    "Provides a reusable list of resource access information.";

  choice transport-access {
    description
      "identify the transport used.";

    case restconf-access {

  }

description
"When the transport is RESTCONF";
leaf-list uri {
  type inet:uri;
  config false;
  ordered-by user;
  description
  "Location of a subscription specific URI on the publisher.";
}

augment "/sn:subscriptions/sn:subscription" {
  description
  "This augmentation allows the message generators to be exposed for a subscription.";
  uses resource-access-list;
  uses message-generator-ids;
}

augment "/sn:subscription-started" {
  description
  "This augmentation allows MSO specific parameters to be exposed for a subscription.";
  uses message-generator-ids;
}

augment "/sn:subscription-modified" {
  description
  "This augmentation allows MSO specific parameters to be exposed for a subscription.";
  uses message-generator-ids;
}

augment "/sn:establish-subscription/sn:output" {
  description
  "This augmentation allows MSO specific parameters to be exposed for a subscription.";
  uses resource-access-list;
  uses message-generator-ids;
}
augment "/sn:modify-subscription/sn:output" {
    description
    "This augmentation allows MSO specific parameters to be exposed for a subscription."

    uses resource-access-list;
    uses message-generator-ids;
}

<CODE ENDS>

10. IANA Considerations

This document registers the following namespace URI in the IETF XML Registry [RFC3688]:


Registrant Contact: The IESG.

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

This document registers the following YANG module in the YANG Module Names registry [RFC3688]:

Name: ietf-multiple-stream-originators


Prefix: mso

Reference: RFC XXXX

11. Transport Considerations

The distributed data export mechanism enabled by this draft is expected to generate more data than YANG-Push. The large amount of data may congest the network and impact other network business. In this case, the collector may also not be able to accept all the data. So the congestion control method is required for any transport that is going to implement the solution proposed in this document.

12. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via 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 (NACM) [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.

The new data nodes introduced 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-config or notification) to this data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

- /subscriptions/subscription/message-generator-ids
- /subscriptions/subscription/resource-access-list

The entries in the two lists above will show where subscribed resources might be located on the publishers. Access control MUST be set so that only someone with proper access permissions has the ability to access this resource.

Other Security Considerations is the same as those discussed in YANG-Push [I-D.ietf-netconf-yang-push].

13. Acknowledgements

TBD

14. References

14.1. Normative References


14.2. Informative References


Appendix A.  Examples

A.1.  RESTCONF Establishing Dynamic Subscription

This example shows how a RESTCONF dynamic subscription is established. The request is given a subscription identifier of 22, and decomposed into two component subscriptions.

Firstly, an establish-subscription request is sent to the Master.

POST /restconf/operations
/ietf-subscribed-notifications:establish-subscription
{
  "ietf-subscribed-notifications:input": {
    "stream-xpath-filter": "/example-module:foo/",
    "stream": "NETCONF",
    "dscp": 10
  }
}

Fig. 4 establish-subscription request

As publisher was able to fully satisfy the request, the Master sends the subscription identifier of the accepted subscription, the URIs, and the message generator IDs:

HTTP status code - 200
{
  "id": 22,
  "uri": [
    "https://192.0.3.1/restconf/subscriptions/22",
    "https://192.0.3.2/restconf/subscriptions/22"
  ],
  "message-generator-id": ["1","2"]
}

Fig. 5 establish-subscription success
Upon receipt of the successful response, the subscriber GET the provided URIs to start the flow of notification messages.

GET https://192.0.3.1/restconf/subscriptions/22
GET https://192.0.3.2/restconf/subscriptions/22

Fig. 6 establish-subscription subsequent POST

A.2. HTTPS Configured Subscription

This example reuses the use case in [I-D.mahesh-netconf-https-notif] and shows how two message originators associated to one subscription can be configured to send https notifications to a receiver at address 192.0.2.1, port 443 with server certificates, and the corresponding trust store that is used to authenticate connections.

[offer: \ line wrapping for formatting only]

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<receiver>
  <name>foo</name>
  <channel>
    <tcp-params>
      <remote-address>192.0.2.1</remote-address>
      <remote-port>443</remote-port>
      <local-address>192.0.3.1</local-address>
      <local-port>63001</local-port>
    </tcp-params>
    <tls-params>
      <server-authentication>
        <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
        <server-certs>explicitly-trusted-server-certs</server-certs>
      </server-authentication>
    </tls-params>
  </channel>
</receiver>
<receiver>
  <name>foo</name>
  <channel>
    <tcp-params>
      <remote-address>192.0.2.1</remote-address>
      <remote-port>443</remote-port>
      <local-address>192.0.3.1</local-address>
      <local-port>63001</local-port>
    </tcp-params>
    <tls-params>
      <server-authentication>
        <ca-certs>explicitly-trusted-server-ca-certs</ca-certs>
        <server-certs>explicitly-trusted-server-certs</server-certs>
      </server-authentication>
    </tls-params>
  </channel>
</receiver>
</receivers>
</config>
<server-authentication>
  <tls-params>
    <channel>
      </receiver>
    </receivers>
</server-authentication>

<subscriptions
  xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
  <subscription>
    <id>6666</id>
    <stream-subtree-filter>foo</stream-subtree-filter>
    <stream>some-stream</stream>
    <receivers>
      <receiver>
        <name>my-receiver1</name>
        <receiver-ref
          foo
        </receiver-ref>
      </receiver>
    </receivers>
  </subscription>
</subscriptions>

<truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
  <certificates>
    <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>
  </certificates>
</truststore>

<certificates>
  <name>explicitly-trusted-server-ca-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>
</certificates>

<certificates>
  <name>Fred Flintstone</name>
  <cert>base64encodedvalue==</cert>
</certificates>

<certificates>
  <name>explicitly-trusted-server-ca-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>
</certificates>
</description>
<certificate>
    <name>ca.example.com</name>
    <cert>base64encodedvalue==</cert>
</certificate>
</certificates>
</truststore>
</config>

Appendix B. Change Log

(To be removed by RFC editor prior to publication)

v01

- Minor revision on Subscription Decomposition
- Revised terminologies
- Removed most implementation related text
- Place holder of two sections: Subscription Management, and Notifications on Subscription State Changes

v02

- Revised section 4 and 5. Moved them from appendix to the main text.

v03

- Added a section for Terminologies.
- Added a section for Subscription State Change Notifications.
- 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.
- Revised the solution overview for a more clear description.

v04

- Added the YANG data model for the proposed augment.

v05
Added the IANA considerations, transport considerations and security considerations.

v06

Added examples.

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
Futurewai
2330 Central Expressway
Santa Clara, California
United States of America

Email: ludwig@clemm.org

Andy Bierman
YumaWorks
United States of America

Email: andy@yumaworks.com