NETCONF Working Group K. Watsen
Internet-Draft Juniper Networks
Intended status: Standards Track H. Wang
Expires: April 25, 2019 Huawei
October 22, 2018

Common YANG Data Types for Cryptography
draft-ietf-netconf-crypto-types-02

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

o "2018-10-22" --> the publication date of this draft

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

o Appendix B. Change Log

Status of This Memo

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 April 25, 2019.

Copyright Notice

Table of Contents

1. Introduction 3
2. The Crypto Types Module 3
  2.1. Tree Diagram 3
  2.2. YANG Module 4
3. Security Considerations 39
4. IANA Considerations 40
  4.1. The IETF XML Registry 40
  4.2. The YANG Module Names Registry 40
5. References 40
  5.1. Normative References 40
  5.2. Informative References 44
Appendix A. Examples 45
  A.1. The "asymmetric-key-pair-with-certs-grouping" Grouping . 45
  A.2. The "generate-hidden-key" Action 47
  A.3. The "install-hidden-key" Action 48
  A.4. The "generate-certificate-signing-request" Action 49
  A.5. The "certificate-expiration" Notification 50
Appendix B. Change Log 51
  B.1. I-D to 00 51
  B.2. 00 to 01 51
  B.3. 01 to 02 51
Acknowledgements 52
Authors' Addresses 52

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.

```
[Note: '\' line wrapping for formatting only]

module: ietf-crypto-types

  grouping asymmetric-key-pair-grouping
    +-- algorithm?             asymmetric-key-encryption-algorithm-r\
ef
    +-- public-key?            binary
    +-- private-key?           union
    +---x generate-hidden-key
    |  +---w input
    |     +---w algorithm    asymmetric-key-encryption-algorithm-ref
    +---x install-hidden-key
       +---w input
          +---w algorithm      asymmetric-key-encryption-algorithm-r\
ef
          +---w public-key?    binary
          +---w private-key?   binary
  grouping public-key-grouping
    +-- algorithm?    asymmetric-key-encryption-algorithm-ref
    +-- public-key?   binary
  grouping asymmetric-key-pair-with-certs-grouping
    +-- algorithm?
    |       asymmetric-key-encryption-algorithm-ref
    +-- public-key?                               binary
    +-- private-key?                              union
    +---x generate-hidden-key
    |  +---w input
```

```
       |     +---w algorithm    asymmetric-key-encryption-algorithm-ref
       +---x install-hidden-key
       |  +---w input
       |     +---w algorithm      asymmetric-key-encryption-algorithm-r\
   ef
       |     +---w public-key?    binary
       |     +---w private-key?   binary
       +-- certificates
       |  +-- certificate* [name]
       |     +-- name?                     string
       |     +-- cert?                     end-entity-cert-cms
       |     +---n certificate-expiration
       |        +-- expiration-date    yang:date-and-time
       +---x generate-certificate-signing-request
          +---w input
          |  +---w subject       binary
          |  +---w attributes?   binary
          +--ro output
             +--ro certificate-signing-request    binary
     grouping end-entity-cert-grouping
       +-- cert?                     end-entity-cert-cms
       +---n certificate-expiration
          +-- expiration-date    yang:date-and-time
     grouping trust-anchor-cert-grouping
       +-- cert?                     trust-anchor-cert-cms
       +---n certificate-expiration
          +-- expiration-date    yang:date-and-time
```

2.2. YANG Module

This module has normative references to [RFC2404], [RFC2986], [RFC3174], [RFC3565], [RFC3686], [RFC4106], [RFC4253], [RFC4279], [RFC4309], [RFC4493], [RFC4494], [RFC4543], [RFC4868], [RFC5280], [RFC5652], [RFC5656], [RFC5915], [RFC6187], [RFC6234], [RFC6239], [RFC6507], [RFC6991], [RFC7539], [RFC7919], [RFC8017], [RFC8032], [RFC8268], [RFC8332], [RFC8341], [RFC8422], [RFC8446], and [ITU.X690.2015].

This module has an informational reference to [RFC6125].

```
<CODE BEGINS> file "ietf-crypto-types@2018-10-22.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:kwatsen@juniper.net>

      Author:   Wang Haiguang
                <wang.haiguang.shieldlab@huawei.com>";

    description
     "This module defines common YANG types for cryptographic
      applications.

      Copyright (c) 2018 IETF Trust and the persons identified
      as authors of the code. All rights reserved.

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

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

    revision "2018-10-22" {
      description
       "Initial version";
      reference
       "RFC XXXX: Common YANG Data Types for Cryptography";
    }
```

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

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

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

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

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

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

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

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

     identity rsa1024 {
       base asymmetric-key-encryption-algorithm;
       description
         "The RSA algorithm using a 1024-bit key.";
       reference
```

```
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }

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

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

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

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

    identity rsa15360 {
      base asymmetric-key-encryption-algorithm;
         description
        "The RSA algorithm using a 15360-bit key.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }
```

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

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

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

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

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

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

    identity hmac-sha2-256-128 {
      base "mac-algorithm";
      description
        "Generating a 256 bits MAC using SHA2 hash function and truncate
         it to 128 bits";
      reference
        "RFC 4868:
           Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with
           IPsec";
```

```
    }

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

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

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

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

    identity aes-128-gmac {
      base "mac-algorithm";
      description
        "Generating MAC using the Advanced Encryption Standard (AES)
         Galois Message Authentication Code (GMAC) as a mechanism to
         provide data origin authentication";
      reference
```

```
         "RFC 4543:
            The Use of Galois Message Authentication Code (GMAC) in
            IPsec ESP and AH";
    }

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

    }

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

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

    identity aes-cmac-128 {
      base "mac-algorithm";
      description
        "Generating MAC using Advanced Encryption Standard (AES)
         Cipher-based Message Authentication Code (CMAC)";
      reference
        "RFC 4493: The AES-CMAC Algorithm";
    }
```

```
     identity mac-aes-128-ccm {
       base "mac-algorithm";
       description
         "Generating MAC using Advanced Encryption Standard (AES) in
          CCM (Counter with CBC-MAC) mode (AES CCM)";
       reference
         "RFC 4309:
            Using Advanced Encryption Standard (AES) CCM Mode with
            IPsec Encapsulating Security Payload (ESP)";
     }

     identity mac-aes-192-ccm {
       base "mac-algorithm";
       description
         "Generating MAC using Advanced Encryption Standard (AES) in
          CCM (Counter with CBC-MAC) mode (AES CCM)";
       reference
         "RFC 4309:
            Using Advanced Encryption Standard (AES) CCM Mode with
            IPsec Encapsulating Security Payload (ESP)";
     }

     identity mac-aes-256-ccm {
       base "mac-algorithm";
       description
         "Generating MAC using Advanced Encryption Standard (AES) in
          CCM (Counter with CBC-MAC) mode (AES CCM)";
       reference
         "RFC 4309:
            Using Advanced Encryption Standard (AES) CCM Mode with
            IPsec Encapsulating Security Payload (ESP)";
     }

     identity mac-aes-128-gcm {
       base "mac-algorithm";
       description
         "Generating MAC when using Advanced Encryption Standard (AES)
          GCM mode for encryption";
       reference
         "RFC 4106:
            The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
            Security Payload (ESP)";
     }

     identity mac-aes-192-gcm {
       base "mac-algorithm";
       description
         "Generating MAC when using Advanced Encryption Standard (AES)
```

```
         GCM mode for encryption";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity mac-aes-256-gcm {
      base "mac-algorithm";
      description
        "Generating MAC when using Advanced Encryption Standard (AES)
         GCM mode for encryption";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity mac-chacha20-poly1305 {
      base "mac-algorithm";
      description
        "Generating MAC using poly1305 algorithm";
      reference
        "RFC 7539: ChaCha20 and Poly1305 for IETF Protocols";
    }


    /********************************************************/
    /*   Identities for Symmetric Key Encryption Algorithms*/
    /********************************************************/

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

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

    identity aes-192-cbc {
```

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

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

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

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

    identity aes-256-ctr {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in CTR mode with a key
         length of 256 bits";
```

```
      reference
        "RFC 3686:
           Using Advanced Encryption Standard (AES) Counter Mode with
           IPsec Encapsulating Security Payload (ESP)";
    }

    identity enc-aes-128-ccm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in CCM mode with a key
         length of 128 bits";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with IPsec
           Encapsulating Security Payload (ESP)";
    }

    identity enc-aes-192-ccm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in CCM mode with a key
         length of 192 bits";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with IPsec
           Encapsulating Security Payload (ESP)";
    }

    identity enc-aes-256-ccm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in CCM mode with a key
         length of 256 bits";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with IPsec
           Encapsulating Security Payload (ESP)";
    }

    identity enc-aes-128-gcm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in GCM mode with a key
         length of 128 bits";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
```

```
    }

    identity enc-aes-192-gcm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in GCM mode with a key
         length of 192 bits";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity enc-aes-256-gcm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in GCM mode with a key
         length of 256 bits";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity enc-chacha20-poly1305 {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with chacha20 algorithm and generate MAC with
         POLY1305";
      reference
        "RFC 7539: ChaCha20 and Poly1305 for IETF Protocols";
    }

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

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

    identity dsa-sha1 {
      base "signature-algorithm";
      description
        "The signature algorithm using DSA algorithm with SHA1 hash
         algorithm";
      reference
```

```
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

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

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

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

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

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

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

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

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

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

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

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

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

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

    identity x509v3-rsa-pkcs1-sha1 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ssh-rsa key format and
         RSASSA-PKCS1-v1_5 with the SHA1 hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-rsa2048-pkcs1-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-rsa2048-sha256
         key format and RSASSA-PKCS1-v1_5 with the SHA-256
         hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-secp256r1-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ecdsa-sha2-secp256r1 key
         format and ECDSA algorithm with the SHA-256 hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-secp384r1-sha384 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ecdsa-sha2-secp384r1 key
         format and ECDSA algorithm with the SHA-384 hash algorithm.";
```

```
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-secp521r1-sha512 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ecdsa-sha2-secp521r1 key
         format and ECDSA algorithm with the SHA-512 hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

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

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

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

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

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

    identity psk-only {
      base "key-exchange-algorithm";
      description
        "Using Pre-shared key for authentication and key exhange";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
          (TLS)";
    }

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

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

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

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

     identity dhe-ffdhe8192 {
       base "key-exchange-algorithm";
       description
         "Ephemeral Diffie Hellman key exhange with 8192 bit
          finite field";
       reference
         "RFC 7919:
            Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
            for Transport Layer Security (TLS)";
     }

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

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

     identity psk-dhe-ffdhe4096 {
       base "key-exchange-algorithm";
       description
         "Key exchange using pre-shared key with Diffie-Hellman key
          generation mechansim, where the DH group is FFDHE4096";
       reference
```

```
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

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

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

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

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

    identity ecdhe-secp521r1 {
      base "key-exchange-algorithm";
```

```
      description
        "Ephemeral Diffie Hellman key exhange with elliptic group
         over curve secp521r1";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity ecdhe-x25519 {
      base "key-exchange-algorithm";
      description
        "Ephemeral Diffie Hellman key exhange with elliptic group
         over curve x25519";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

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

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

    identity psk-ecdhe-secp384r1 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with elliptic group-based
         Ephemeral Diffie Hellman key exhange over curve secp384r1";
      reference
        "RFC 8446:
```

```
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

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

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

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

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

   identity diffie-hellman-group14-sha256 {
      base "key-exchange-algorithm";
      description
        "Using DH group14 and SHA256 for key exchange";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
```

```
            Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

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

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

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

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

    identity ecdh-sha2-secp256r1 {
      base "key-exchange-algorithm";
      description
        "Elliptic curve-based Diffie Hellman key exhange over curve
         secp256r1 and using SHA2 for MAC generation";
```

```
      reference
        "RFC 6239: Suite B Cryptographic Suites for Secure Shell (SSH)";
    }

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

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

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

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

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

    typedef symmetric-key-encryption-algorithm-ref {
      type identityref {
        base "symmetric-key-encryption-algorithm";
      }
```

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

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

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

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

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

     typedef crl {
       type binary;
```

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

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

    typedef cms {
      type binary;
      description
        "A ContentInfo structure, as specified in RFC 5652,
         encoded using ASN.1 distinguished encoding rules (DER),
         as specified in ITU-T X.690.";
      reference
        "RFC 5652:
           Cryptographic Message Syntax (CMS)
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }

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

    typedef signed-data-cms {
      type cms;
      description
        "A CMS structure whose top-most content type MUST be the
```

```
         signed-data content type, as described by Section 5 in
         RFC 5652.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
    }

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

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

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

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

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

```
    /*************************************************/

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

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

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

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

    typedef end-entity-cert-x509 {
      type x509;
      description
        "A Certificate structure that MUST encode a certificate
         that is neither self-signed nor having Basic constraint
         CA true.";
    }

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

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

         The CMS MUST contain only a single chain of certificates.
         The client or end-entity certificate MUST only authenticate
```

```
         to last intermediate CA certificate listed in the chain.

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

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

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

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

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

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

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

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

    grouping public-key-grouping {
      description
        "A public key.";
      leaf algorithm {
        type asymmetric-key-encryption-algorithm-ref;
        description
          "Identifies the key's algorithm.  More specifically,
           this leaf specifies how the 'public-key' binary leaf
           is encoded.";
        reference
          "RFC CCCC: Common YANG Data Types for Cryptography";
      }
      leaf public-key {
        type binary;
        description
          "A binary that contains the value of the public key.  The
           interpretation of the content is defined by the key
           algorithm.  For example, a DSA key is an integer, an RSA
           key is represented as RSAPublicKey as defined in
           RFC 8017, and an Elliptic Curve Cryptography (ECC) key
           is represented using the 'publicKey' described in
           RFC 5915.";
        reference
          "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                     RSA Cryptography Specifications Version 2.2.
           RFC 5915: Elliptic Curve Private Key Structure.";
      }
    } // end public-key-grouping


    grouping asymmetric-key-pair-grouping {
      description
        "A private/public key pair.";
      uses public-key-grouping;
      leaf private-key {
        nacm:default-deny-all;
        type union {
          type binary;
          type enumeration {
            enum "permanently-hidden" {
              description
               "The private key is inaccessible due to being
                protected by the system (e.g., a cryptographic
                hardware module).  It is not possible to
```

```
                configure a permanently hidden key, as a real
                private key value must be set.  Permanently
                hidden keys cannot be archived or backed up.";
            }
          }
        }
        description
          "A binary that contains the value of the private key.  The
           interpretation of the content is defined by the key
           algorithm.  For example, a DSA key is an integer, an RSA
           key is represented as RSAPrivateKey as defined in
           RFC 8017, and an Elliptic Curve Cryptography (ECC) key
           is represented as ECPrivateKey as defined in RFC 5915.";
        reference
          "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                     RSA Cryptography Specifications Version 2.2.
           RFC 5915: Elliptic Curve Private Key Structure.";
      } // end private-key

      action generate-hidden-key {
        description
          "Requests the device to generate a hidden key using the
           specified asymmetric key algorithm.  This action is
           used to request the system to generate a key that
           is 'permanently-hidden', perhaps protected by a
           cryptographic hardware module.  The resulting
           asymmetric key values are considered operational
           state and hence present only in <operational>.";
        input {
          leaf algorithm {
            type asymmetric-key-encryption-algorithm-ref;
            mandatory true;
            description
              "The algorithm to be used when generating the
               asymmetric key.";
            reference
              "RFC CCCC: Common YANG Data Types for Cryptography";
          }
        }
      } // end generate-hidden-key

      action install-hidden-key {
        description
          "Requests the device to load the specified values into
           a hidden key.  The resulting asymmetric key values are
           considered operational state and hence present only in
           <operational>.";
        input {
```

```
          leaf algorithm {
            type asymmetric-key-encryption-algorithm-ref;
            mandatory true;
            description
              "The algorithm to be used when generating the
               asymmetric key.";
            reference
              "RFC CCCC: Common YANG Data Types for Cryptography";
          }
          leaf public-key {
            type binary;
            description
              "A binary that contains the value of the public key.
               The interpretation of the content is defined by the key
               algorithm.  For example, a DSA key is an integer, an
               RSA key is represented as RSAPublicKey as defined in
               RFC 8017, and an Elliptic Curve Cryptography (ECC) key
               is represented using the 'publicKey' described in
               RFC 5915.";
            reference
              "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                         RSA Cryptography Specifications Version 2.2.
               RFC 5915: Elliptic Curve Private Key Structure.";
          }
          leaf private-key {
            type binary;
            description
              "A binary that contains the value of the private key.
               The interpretation of the content is defined by the key
               algorithm.  For example, a DSA key is an integer, an RSA
               key is represented as RSAPrivateKey as defined in
               RFC 8017, and an Elliptic Curve Cryptography (ECC) key
               is represented as ECPrivateKey as defined in RFC 5915.";
            reference
              "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                         RSA Cryptography Specifications Version 2.2.
               RFC 5915: Elliptic Curve Private Key Structure.";
          }
        }
      } // end install-hidden-key
    } // end asymmetric-key-pair-grouping


    grouping trust-anchor-cert-grouping {
      description
       "A certificate, and a notification for when it might expire.";
      leaf cert {
        type trust-anchor-cert-cms;
```

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


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

```
    } // end end-entity-cert-grouping


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

          }
          uses end-entity-cert-grouping;
        } // end certificate
      } // end certificates

      action generate-certificate-signing-request {
        description
          "Generates a certificate signing request structure for
           the associated asymmetric key using the passed subject
           and attribute values.  The specified assertions need
           to be appropriate for the certificate's use.  For
           example, an entity certificate for a TLS server
           SHOULD have values that enable clients to satisfy
           RFC 6125 processing.";
        input {
          leaf subject {
            type binary;
            mandatory true;
            description
             "The 'subject' field per the CertificationRequestInfo
               structure as specified by RFC 2986, Section 4.1
               encoded using the ASN.1 distinguished encoding
               rules (DER), as specified in ITU-T X.690.";
```

```
            reference
              "RFC 2986:
                 PKCS #10: Certification Request Syntax
                           Specification Version 1.7.
               ITU-T X.690:
                 Information technology - ASN.1 encoding rules:
                 Specification of Basic Encoding Rules (BER),
                 Canonical Encoding Rules (CER) and Distinguished
                 Encoding Rules (DER).";
          }
          leaf attributes {
            type binary;
            description
              "The 'attributes' field from the structure
               CertificationRequestInfo as specified by RFC 2986,
               Section 4.1 encoded using the ASN.1 distinguished
               encoding rules (DER), as specified in ITU-T X.690.";
            reference
              "RFC 2986:
                 PKCS #10: Certification Request Syntax
                           Specification Version 1.7.
               ITU-T X.690:
                 Information technology - ASN.1 encoding rules:
                 Specification of Basic Encoding Rules (BER),
                 Canonical Encoding Rules (CER) and Distinguished
                 Encoding Rules (DER).";
          }
        }
        output {
          leaf certificate-signing-request {
            type binary;
            mandatory true;
            description
              "A CertificationRequest structure as specified by
               RFC 2986, Section 4.2 encoded using the ASN.1
               distinguished encoding rules (DER), as specified
               in ITU-T X.690.";
            reference
              "RFC 2986:
                 PKCS #10: Certification Request Syntax
                           Specification Version 1.7.
               ITU-T X.690:
                 Information technology - ASN.1 encoding rules:
                 Specification of Basic Encoding Rules (BER),
                 Canonical Encoding Rules (CER) and Distinguished
                 Encoding Rules (DER).";

          }
```

```
        }
      } // end generate-certificate-signing-request
    } // end asymmetric-key-pair-with-certs-grouping


  }
  <CODE ENDS>
```

3. Security Considerations

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

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

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

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

This document uses PKCS #10 [RFC2986] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [RFC4211] was considered, but is was unclear if there was market demand for it. If it is desired to support CRMF in the future, placing a "choice" statement in both the input and output statements, along with an "if-feature" statement on the CRMF option, would enable a backwards compatible solution.

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

4. IANA Considerations

4.1. The IETF XML Registry

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

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

4.2. The YANG Module Names Registry

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

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

5. References

5.1. Normative References

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

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

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, DOI 10.17487/RFC2404, November 1998, <https://www.rfc-editor.org/info/rfc2404>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, DOI 10.17487/RFC3565, July 2003, <https://www.rfc-editor.org/info/rfc3565>.

[RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <https://www.rfc-editor.org/info/rfc3686>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <https://www.rfc-editor.org/info/rfc4106>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <https://www.rfc-editor.org/info/rfc4309>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <https://www.rfc-editor.org/info/rfc4493>.

[RFC4494] Song, JH., Poovendran, R., and J. Lee, "The AES-CMAC-96 Algorithm and Its Use with IPsec", RFC 4494, DOI 10.17487/RFC4494, June 2006, <https://www.rfc-editor.org/info/rfc4494>.

[RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <https://www.rfc-editor.org/info/rfc4543>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <https://www.rfc-editor.org/info/rfc4868>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", RFC 5915, DOI 10.17487/RFC5915, June 2010, <https://www.rfc-editor.org/info/rfc5915>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6239] Igoe, K., "Suite B Cryptographic Suites for Secure Shell (SSH)", RFC 6239, DOI 10.17487/RFC6239, May 2011, <https://www.rfc-editor.org/info/rfc6239>.

[RFC6507] Groves, M., "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)", RFC 6507, DOI 10.17487/RFC6507, February 2012, <https://www.rfc-editor.org/info/rfc6507>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015, <https://www.rfc-editor.org/info/rfc7539>.

[RFC7919] Gillmor, D., "Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)", RFC 7919, DOI 10.17487/RFC7919, August 2016, <https://www.rfc-editor.org/info/rfc7919>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

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

[RFC8268] Baushke, M., "More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)", RFC 8268, DOI 10.17487/RFC8268, December 2017, <https://www.rfc-editor.org/info/rfc8268>.

[RFC8332] Bider, D., "Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol", RFC 8332, DOI 10.17487/RFC8332, March 2018, <https://www.rfc-editor.org/info/rfc8332>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

5.2. Informative References

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

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <https://www.rfc-editor.org/info/rfc5056>.

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

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Examples

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

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

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

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

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

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

     organization
      "Example Corporation";

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

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

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

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

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


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


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

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

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

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

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

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

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

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

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

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

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

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

A.5. The "certificate-expiration" Notification

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

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


Appendix B. Change Log

B.1. I-D to 00

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

B.2. 00 to 01

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

B.3. 01 to 02

- Removed unwanted "mandatory" and "must" statements.
- Added many new crypto algorithms (thanks Haiguang!)
- 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).

Acknowledgements

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

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Wang Haiguang
Huawei

EMail: wang.haiguang.shieldlab@huawei.com

NETCONF Working Group K. Watsen
Internet-Draft Juniper Networks
Intended status: Standards Track October 22, 2018
Expires: April 25, 2019

YANG Data Model for a Centralized Keystore Mechanism
draft-ietf-netconf-keystore-07

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

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

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

o "2018-10-22" --> the publication date of this draft

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

o Appendix A. Change Log

Status of This Memo

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 April 25, 2019.

Copyright Notice

Table of Contents

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

1. Introduction

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

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

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

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

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

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

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

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP

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

3. The Keystore Model

3.1. Tree Diagram

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

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

  grouping local-or-keystore-end-entity-cert-with-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-keys-supported}?
       |  +-- algorithm?
       |  |       asymmetric-key-encryption-algorithm-ref
```

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

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

3.2. Example Usage

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

```
[Note: '\' line wrapping for formatting only]

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

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

```
            <name>ex-rsa-cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>

   <!-- waiting for Haiguang fix...
      <asymmetric-key>
        <name>tls-ec-key</name>
        <algorithm>ct:secp256r1</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <certificates>
          <certificate>
            <name>tls-ec-cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>
   -->

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

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

```
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>

    </asymmetric-keys>
  </keystore>
```

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

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

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

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

  organization
   "Example Corporation";

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

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

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

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

    list just-a-key {
      key name;
```

```
         leaf name {
           type string;
           description
             "An arbitrary name for this key.";
         }
         uses ks:local-or-keystore-asymmetric-key-grouping;
         description
           "An asymmetric key, with no certs, that may be configured
            locally or be a reference to an asymmetric key in the
            keystore.  The intent is to reference just the asymmetric
            key, not any certificates that may also be associated
            with the asymmetric key.";
       }

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

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

   }
```

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

[Note: '\' line wrapping for formatting only]


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

  <just-a-key>
    <name>a keystore-defined key (and its associated certs)</name>
    <reference>ex-rsa-key</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>
    <algorithm
      xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
      ct:rsa2048
    </algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <certificates>
      <certificate>
        <name>a locally-defined cert</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </key-with-certs>

  <key-with-certs>
    <name>a keystore-defined key (and its associated certs)</name>
    <reference>ex-rsa-key</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>
       <algorithm
         xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
         ct:rsa2048
       </algorithm>
       <private-key>base64encodedvalue==</private-key>
       <public-key>base64encodedvalue==</public-key>
       <cert>base64encodedvalue==</cert>
     </end-entity-cert-with-key>

     <end-entity-cert-with-key>
       <name>a keystore-defined certificate (and its associated key)</n\
   ame>
       <reference>ex-rsa-cert</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@2018-10-22.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:kwatsen@juniper.net>";


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

       Copyright (c) 2018 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC VVVV:
           YANG Data Model for a Centralized Keystore Mechanism";
     }

     // Features

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

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

     // Typedefs
```

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

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


     // Groupings

     grouping local-or-keystore-asymmetric-key-grouping {
       description
         "A grouping that expands to allow the asymmetric key to be
          either stored locally, within the using data model, or be
          a reference to an asymmetric key stored in the keystore.";
       choice local-or-keystore {
         mandatory true;
         case local {
           if-feature "local-keys-supported";
           uses ct:asymmetric-key-pair-grouping;
         }
         case keystore {
           if-feature "keystore-supported";
           leaf reference {
             type ks:asymmetric-key-ref;
             description
               "A reference to an asymmetric key that exists in
                the keystore.  The intent is to reference just the
                asymmetric key, not any certificates that may also
                be associated with the asymmetric key.";
```

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

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

    grouping local-or-keystore-end-entity-cert-with-key-grouping {
      description
        "A grouping that expands to allow an end-entity certificate
         (and its associated private key) to be either stored locally,
         within the using data model, or be a reference to a specific
         certificate in the keystore.";
      choice local-or-keystore {
        mandatory true;
        case local {
          if-feature "local-keys-supported";
          uses ct:asymmetric-key-pair-grouping;
          uses ct:end-entity-cert-grouping;
        }
        case keystore {
```

```
          if-feature "keystore-supported";
          leaf reference {
            type ks:asymmetric-key-certificate-ref;
            description
              "A reference to a specific certificate, and its
               associated private key, stored in the keystore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
      }
    }



    // protocol accessible nodes

    container keystore {
      nacm:default-deny-write;

      description
        "The keystore contains a list of keys.";

      container asymmetric-keys {
        description
          "A list of asymmetric keys.";
        list asymmetric-key {
          must "(algorithm and public-key and private-key)
                or not (algorithm or public-key or private-key)";
          key name;
          description
            "An asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the asymmetric key.  If the name
               matches the name of a key that exists independently in
               <operational> (i.e., a 'permanently-hidden' key), then
               the 'algorithm', 'public-key', and 'private-key' nodes
               MUST NOT be configured.";
          }
          uses ct:asymmetric-key-pair-with-certs-grouping;
        } // end asymmetric-key

      } // end asymmetric-keys
    } // end keystore
```

```
   }
   <CODE ENDS>
```

4. Security Considerations

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

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

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

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

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

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

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

5. IANA Considerations

5.1. The IETF XML Registry

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

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

5.2. The YANG Module Names Registry

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

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

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-01 (work in progress), September 2018.

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

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

6.2. Informative References

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

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

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[Std-802.1AR-2009]
IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

Appendix A. Change Log

A.1. 00 to 01

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

A.2. 01 to 02

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

A.3. 02 to 03

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

A.4. 03 to 04

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

A.5. 04 to 05

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

A.6. 05 to 06

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

A.7. 06 to 07

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

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, Mahesh Jethanandani, Radek Krejci, Reshad Rahman, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Eric Voit, Bert Wijnen, and Liang Xia.

Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

K. Watsen
Juniper Networks
October 22, 2018

NETCONF Client and Server Models
draft-ietf-netconf-netconf-client-server-08

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

o I-D.ietf-netconf-keystore

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

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

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

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

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

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

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

o "2018-10-22" --> the publication date of this draft

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

o Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction   3
2. Terminology   4
3. The NETCONF Client Model   4
  3.1. Tree Diagram   4
  3.2. Example Usage 12
  3.3. YANG Module 14
4. The NETCONF Server Model 24
  4.1. Tree Diagram 25

4.2. Example Usage 32
4.3. YANG Module 37
5. Design Considerations 49
5.1. Support all NETCONF transports 49
5.2. Enable each transport to select which keys to use 49
5.3. Support authenticating NETCONF clients certificates . . . 49
5.4. Support mapping authenticated NETCONF client certificates to usernames 50
5.5. Support both listening for connections and call home . . 50
5.6. For Call Home connections 50
5.6.1. Support more than one NETCONF client 50
5.6.2. Support NETCONF clients having more than one endpoint 50
5.6.3. Support a reconnection strategy 50
5.6.4. Support both persistent and periodic connections . . 51
5.6.5. Reconnection strategy for periodic connections . . . 51
5.6.6. Keep-alives for persistent connections 51
5.6.7. Customizations for periodic connections 51
6. Security Considerations 51
7. IANA Considerations 52
7.1. The IETF XML Registry 52
7.2. The YANG Module Names Registry 53
8. References 53
8.1. Normative References 53
8.2. Informative References 54
Appendix A. Change Log 56
A.1. 00 to 01 56
A.2. 01 to 02 56
A.3. 02 to 03 56
A.4. 03 to 04 56
A.5. 04 to 05 56
A.6. 05 to 06 57
A.7. 06 to 07 57
A.8. 07 to 08 57
Acknowledgements 57
Author's Address 57

1. Introduction

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

2. Terminology

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

3. The NETCONF Client Model

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

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

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

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

3.1. Tree Diagram

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

```
[Note: '\' line wrapping for formatting only]

module: ietf-netconf-client
  +--rw netconf-client
     +--rw initiate! {initiate}?
     |  +--rw netconf-server* [name]
     |     +--rw name                  string
     |     +--rw endpoints
     |     |  +--rw endpoint* [name]
     |     |     +--rw name        string
     |     |     +--rw (transport)
     |     |        +--:(ssh) {ssh-initiate}?
     |     |        |  +--rw ssh
     |     |        |     +--rw address?           inet:host
     |     |        |     +--rw port?              inet:port-number
```

```
        |     |        |     +--rw client-identity
        |     |        |     |  +--rw username?            string
        |     |        |     |  +--rw (auth-type)
        |     |        |     |     +--:(password)
        |     |        |     |     |  +--rw password?      string
        |     |        |     |     +--:(public-key)
        |     |        |     |     |  +--rw public-key
        |     |        |     |     |     +--rw (local-or-keystore)
        |     |        |     |     |        +--:(local)
        |     |        |     |     |        |        {local-keys-suppor\
   ted}?
        |     |        |     |     |        |  +--rw algorithm?
        |     |        |     |     |        |  |       asymmetric-key-e\
   ncryption-algorithm-ref
        |     |        |     |     |        |  +--rw public-key?
        |     |        |     |     |        |  |       binary
        |     |        |     |     |        |  +--rw private-key?
        |     |        |     |     |        |  |       union
        |     |        |     |     |        |  +---x generate-hidden-key
        |     |        |     |     |        |  |  +---w input
        |     |        |     |     |        |  |     +---w algorithm
        |     |        |     |     |        |  |             asymmetric\
   -key-encryption-algorithm-ref
        |     |        |     |     |        |  +---x install-hidden-key
        |     |        |     |     |        |     +---w input
        |     |        |     |     |        |        +---w algorithm
        |     |        |     |     |        |        |       asymmetric\
   -key-encryption-algorithm-ref
        |     |        |     |     |        |        +---w public-key?
        |     |        |     |     |        |        |       binary
        |     |        |     |     |        |        +---w private-key?
        |     |        |     |     |        |                binary
        |     |        |     |     |        +--:(keystore)
        |     |        |     |     |                 {keystore-supporte\
   d}?
        |     |        |     |     |           +--rw reference?
        |     |        |     |     |                   ks:asymmetric-ke\
   y-ref
        |     |        |     |     +--:(certificate)
        |     |        |     |        +--rw certificate
        |     |        |     |                {sshcmn:ssh-x509-certs}?
        |     |        |     |           +--rw (local-or-keystore)
        |     |        |     |              +--:(local)
        |     |        |     |              |        {local-keys-suppor\
   ted}?
        |     |        |     |              |  +--rw algorithm?
        |     |        |     |              |  |       asymmetric-key-e\
   ncryption-algorithm-ref
```

```
     |     |        |     |              |  +--rw public-key?
     |     |        |     |              |  |       binary
     |     |        |     |              |  +--rw private-key?
     |     |        |     |              |  |       union
     |     |        |     |              |  +---x generate-hidden-key
     |     |        |     |              |  |  +---w input
     |     |        |     |              |  |     +---w algorithm
     |     |        |     |              |  |             asymmetric\
-key-encryption-algorithm-ref
     |     |        |     |              |  +---x install-hidden-key
     |     |        |     |              |  |  +---w input
     |     |        |     |              |  |     +---w algorithm
     |     |        |     |              |  |     |       asymmetric\
-key-encryption-algorithm-ref
     |     |        |     |              |  |     +---w public-key?
     |     |        |     |              |  |     |       binary
     |     |        |     |              |  |     +---w private-key?
     |     |        |     |              |  |             binary
     |     |        |     |              |  +--rw cert?
     |     |        |     |              |  |       end-entity-cert-\
cms
     |     |        |     |              |  +---n certificate-expira\
tion
     |     |        |     |              |     +-- expiration-date
     |     |        |     |              |             yang:date-and\
-time
     |     |        |     |              +--:(keystore)
     |     |        |     |                       {keystore-supporte\
d}?
     |     |        |     |                 +--rw reference?
     |     |        |     |                         ks:asymmetric-ke\
y-certificate-ref
     |     |        |     +--rw server-auth
     |     |        |     |  +--rw pinned-ssh-host-keys?
     |     |        |     |  |       ta:pinned-host-keys-ref
     |     |        |     |  |       {ta:ssh-host-keys}?
     |     |        |     |  +--rw pinned-ca-certs?
     |     |        |     |  |       ta:pinned-certificates-ref
     |     |        |     |  |       {sshcmn:ssh-x509-certs,ta:x509-\
certificates}?
     |     |        |     |  +--rw pinned-server-certs?
     |     |        |     |          ta:pinned-certificates-ref
     |     |        |     |          {sshcmn:ssh-x509-certs,ta:x509-\
certificates}?
     |     |        |     +--rw transport-params
     |     |        |             {ssh-client-transport-params-confi\
g}?
     |     |        |        +--rw host-key
```

```
        |     |        |        |  +--rw host-key-alg*   identityref
        |     |        |        +--rw key-exchange
        |     |        |        |  +--rw key-exchange-alg*   identityref
        |     |        |        +--rw encryption
        |     |        |        |  +--rw encryption-alg*   identityref
        |     |        |        +--rw mac
        |     |        |           +--rw mac-alg*   identityref
        |     |        +--:(tls) {tls-initiate}?
        |     |           +--rw tls
        |     |              +--rw address?           inet:host
        |     |              +--rw port?              inet:port-number
        |     |              +--rw client-identity
        |     |              |  +--rw (auth-type)
        |     |              |     +--:(certificate)
        |     |              |        +--rw certificate
        |     |              |           +--rw (local-or-keystore)
        |     |              |              +--:(local)
        |     |              |              |        {local-keys-suppor\
   ted}?
        |     |              |              |  +--rw algorithm?
        |     |              |              |  |       asymmetric-key-e\
   ncryption-algorithm-ref
        |     |              |              |  +--rw public-key?
        |     |              |              |  |       binary
        |     |              |              |  +--rw private-key?
        |     |              |              |  |       union
        |     |              |              |  +---x generate-hidden-key
        |     |              |              |  |  +---w input
        |     |              |              |  |     +---w algorithm
        |     |              |              |  |             asymmetric\
   -key-encryption-algorithm-ref
        |     |              |              |  +---x install-hidden-key
        |     |              |              |  |  +---w input
        |     |              |              |  |     +---w algorithm
        |     |              |              |  |     |       asymmetric\
   -key-encryption-algorithm-ref
        |     |              |              |  |     +---w public-key?
        |     |              |              |  |     |       binary
        |     |              |              |  |     +---w private-key?
        |     |              |              |  |             binary
        |     |              |              |  +--rw cert?
        |     |              |              |  |       end-entity-cert-\
   cms
        |     |              |              |  +---n certificate-expira\
   tion
        |     |              |              |     +-- expiration-date
        |     |              |              |             yang:date-and\
   -time
```

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

```
                 |     |     +--:(public-key)
                 |     |     |  +--rw public-key
                 |     |     |     +--rw (local-or-keystore)
                 |     |     |        +--:(local) {local-keys-supported\
   }?
                 |     |     |        |  +--rw algorithm?
                 |     |     |        |  |       asymmetric-key-encrypt\
   ion-algorithm-ref
                 |     |     |        |  +--rw public-key?
                 |     |     |        |  |       binary
                 |     |     |        |  +--rw private-key?
                 |     |     |        |  |       union
                 |     |     |        |  +---x generate-hidden-key
                 |     |     |        |  |  +---w input
                 |     |     |        |  |     +---w algorithm
                 |     |     |        |  |             asymmetric-key-e\
   ncryption-algorithm-ref
                 |     |     |        |  +---x install-hidden-key
                 |     |     |        |     +---w input
                 |     |     |        |        +---w algorithm
                 |     |     |        |        |       asymmetric-key-e\
   ncryption-algorithm-ref
                 |     |     |        |        +---w public-key?    bin\
   ary
                 |     |     |        |        +---w private-key?   bin\
   ary
                 |     |     |        +--:(keystore) {keystore-supporte\
   d}?
                 |     |     |           +--rw reference?
                 |     |     |                   ks:asymmetric-key-ref
                 |     |     +--:(certificate)
                 |     |        +--rw certificate {sshcmn:ssh-x509-cert\
   s}?
                 |     |           +--rw (local-or-keystore)
                 |     |              +--:(local) {local-keys-supported\
   }?
                 |     |              |  +--rw algorithm?
                 |     |              |  |       asymmetric-key-encrypt\
   ion-algorithm-ref
                 |     |              |  +--rw public-key?
                 |     |              |  |       binary
                 |     |              |  +--rw private-key?
                 |     |              |  |       union
                 |     |              |  +---x generate-hidden-key
                 |     |              |  |  +---w input
                 |     |              |  |     +---w algorithm
                 |     |              |  |             asymmetric-key-e\
   ncryption-algorithm-ref
```

```
                |     |              |  +---x install-hidden-key
                |     |              |  |  +---w input
                |     |              |  |     +---w algorithm
                |     |              |  |     |       asymmetric-key-e\
ncryption-algorithm-ref
                |     |              |  |     +---w public-key?    bin\
ary
                |     |              |  |     +---w private-key?   bin\
ary
                |     |              |  +--rw cert?
                |     |              |  |       end-entity-cert-cms
                |     |              |  +---n certificate-expiration
                |     |              |     +-- expiration-date
                |     |              |             yang:date-and-time
                |     |              +--:(keystore) {keystore-supporte\
d}?
                |     |                 +--rw reference?
                |     |                         ks:asymmetric-key-cert\
ificate-ref
                |     +--rw server-auth
                |     |  +--rw pinned-ssh-host-keys?
                |     |  |       ta:pinned-host-keys-ref
                |     |  |       {ta:ssh-host-keys}?
                |     |  +--rw pinned-ca-certs?
                |     |  |       ta:pinned-certificates-ref
                |     |  |       {sshcmn:ssh-x509-certs,ta:x509-certif\
icates}?
                |     |  +--rw pinned-server-certs?
                |     |          ta:pinned-certificates-ref
                |     |          {sshcmn:ssh-x509-certs,ta:x509-certif\
icates}?
                |     +--rw transport-params
                |             {ssh-client-transport-params-config}?
                |        +--rw host-key
                |        |  +--rw host-key-alg*   identityref
                |        +--rw key-exchange
                |        |  +--rw key-exchange-alg*   identityref
                |        +--rw encryption
                |        |  +--rw encryption-alg*   identityref
                |        +--rw mac
                |           +--rw mac-alg*   identityref
                +--:(tls) {tls-listen}?
                   +--rw tls
                      +--rw address?          inet:ip-address
                      +--rw port?             inet:port-number
                      +--rw client-identity
                      |  +--rw (auth-type)
                      |     +--:(certificate)
```

```
                       |        +--rw certificate
                       |           +--rw (local-or-keystore)
                       |              +--:(local) {local-keys-supported\
   }?
                       |              |  +--rw algorithm?
                       |              |  |       asymmetric-key-encrypt\
   ion-algorithm-ref
                       |              |  +--rw public-key?
                       |              |  |       binary
                       |              |  +--rw private-key?
                       |              |  |       union
                       |              |  +---x generate-hidden-key
                       |              |  |  +---w input
                       |              |  |     +---w algorithm
                       |              |  |             asymmetric-key-e\
   ncryption-algorithm-ref
                       |              |  +---x install-hidden-key
                       |              |  |  +---w input
                       |              |  |     +---w algorithm
                       |              |  |     |       asymmetric-key-e\
   ncryption-algorithm-ref
                       |              |  |     +---w public-key?    bin\
   ary
                       |              |  |     +---w private-key?   bin\
   ary
                       |              |  +--rw cert?
                       |              |  |       end-entity-cert-cms
                       |              |  +---n certificate-expiration
                       |              |     +-- expiration-date
                       |              |             yang:date-and-time
                       |              +--:(keystore) {keystore-supporte\
   d}?
                       |                 +--rw reference?
                       |                         ks:asymmetric-key-cert\
   ificate-ref
                       +--rw server-auth
                       |  +--rw pinned-ca-certs?
                       |  |       ta:pinned-certificates-ref
                       |  |       {ta:x509-certificates}?
                       |  +--rw pinned-server-certs?
                       |          ta:pinned-certificates-ref
                       |          {ta:x509-certificates}?
                       +--rw hello-params
                               {tls-client-hello-params-config}?
                          +--rw tls-versions
                          |  +--rw tls-version*   identityref
                          +--rw cipher-suites
                             +--rw cipher-suite*   identityref
```

3.2. Example Usage

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

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

[Note: '\' line wrapping for formatting only]

```
<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>
            <address>corp-fw1.example.com</address>
            <client-identity>
              <username>foobar</username>
              <public-key>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
f-crypto-types">ct:rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
              </public-key>
            </client-identity>
            <server-auth>
              <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
inned-ca-certs>
              <pinned-server-certs>explicitly-trusted-server-certs</\
pinned-server-certs>
            </server-auth>
          </ssh>
        </endpoint>
        <endpoint>
          <name>corp-fw2.example.com</name>
          <ssh>
            <address>corp-fw2.example.com</address>
            <client-identity>
              <username>foobar</username>
```

```
                <public-key>
                  <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
   f-crypto-types">ct:rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                </public-key>
              </client-identity>
              <server-auth>
                <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
   inned-ca-certs>
                <pinned-server-certs>explicitly-trusted-server-certs</\
   pinned-server-certs>
              </server-auth>
            </ssh>
          </endpoint>
        </endpoints>
        <connection-type>
          <persistent/>
        </connection-type>
        <reconnect-strategy>
          <start-with>last-connected</start-with>
        </reconnect-strategy>
      </netconf-server>
    </initiate>

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

```
        </ssh>
      </endpoint>
    </listen>
  </netconf-client>
```

3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-netconf-client@2018-10-22.yang"
module ietf-netconf-client {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-client";
  prefix "ncc";

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

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

  import ietf-ssh-client {
    prefix ss;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC YYYY: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-tls-client {
    prefix ts;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
```

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

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>

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


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

       Copyright (c) 2017 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: NETCONF Client and Server Models";
     }

     // Features

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

     feature ssh-initiate {
       description
        "The 'ssh-initiate' feature indicates that the NETCONF client
         supports initiating SSH connections to NETCONF servers.";
       reference
```

```
        "RFC 6242:
           Using the NETCONF Protocol over Secure Shell (SSH)";
     }

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

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

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

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

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

     grouping netconf-client-grouping {
       description
```

```
         "Top-level grouping for NETCONF client configuration.";

       container initiate {
         if-feature initiate;
         presence "Enables client to initiate TCP connections";
         description
           "Configures client initiating underlying TCP connections.";
         list netconf-server {
           key name;
           min-elements 1;
           description
             "List of NETCONF servers the NETCONF client is to
              initiate connections to in parallel.";
           leaf name {
             type string;
             description
               "An arbitrary name for the NETCONF server.";
           }
           container endpoints {
             description
               "Container for the list of endpoints.";
             list endpoint {
               key name;
               min-elements 1;
               ordered-by user;
               description
                 "A user-ordered list of endpoints that the NETCONF
                  client will attempt to connect to in the specified
                  sequence.  Defining more than one enables
                  high-availability.";
               leaf name {
                 type string;
                 description
                   "An arbitrary name for the endpoint.";
               }
               choice transport {
                 mandatory true;
                 description
                   "Selects between available transports.";
                 case ssh {
                   if-feature ssh-initiate;
                   container ssh {
                     description
                       "Specifies IP and SSH specific configuration
                        for the connection.";
                     leaf address {
                       type inet:host;
                       description
```

```
                    "The IP address or hostname of the endpoint.
                     If a domain name is configured, then the
                     DNS resolution should happen on each usage
                     attempt.  If the DNS resolution results in
                     multiple IP addresses, the IP addresses will
                     be tried according to local preference order
                     until a connection has been established or
                     until all IP addresses have failed.";
                }
                leaf port {
                  type inet:port-number;
                  default 830;
                  description
                    "The IP port for this endpoint.  The NETCONF
                     client will use the IANA-assigned well-known
                     port for 'netconf-ssh' (830) if no value is
                     specified.";
                }
                uses ss:ssh-client-grouping;
              }
            } // end ssh
            case tls {
              if-feature tls-initiate;
              container tls {
                description
                  "Specifies IP and TLS specific configuration
                   for the connection.";
                leaf address {
                  type inet:host;
                  description
                    "The IP address or hostname of the endpoint.
                     If a domain name is configured, then the
                     DNS resolution should happen on each usage
                     attempt.  If the DNS resolution results in
                     multiple IP addresses, the IP addresses will
                     be tried according to local preference order
                     until a connection has been established or
                     until all IP addresses have failed.";
                  }
                  leaf port {
                    type inet:port-number;
                    default 6513;
                    description
                      "The IP port for this endpoint. The NETCONF
                       client will use the IANA-assigned well-
                       known port for 'netconf-tls' (6513) if no
                       value is specified.";
                  }
```

```
                  uses ts:tls-client-grouping {
                    refine "client-identity/auth-type" {
                    mandatory true;
                     description
                       "NETCONF/TLS clients MUST pass some
                        authentication credentials.";
                   }
                 }
               }
             } // end tls
           }
         }
       }

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

                 This connection type minimizes any NETCONF server
                 to NETCONF client data-transfer delay, albeit at
                 the expense of holding resources longer.";
               container keep-alives {
                 description
                   "Configures the keep-alive policy, to
                    proactively test the aliveness of the SSH/TLS
                    server.  An unresponsive SSH/TLS server will
                    be dropped after approximately max-attempts *
                    max-wait seconds.";
                 leaf max-wait {
                   type uint16 {
                     range "1..max";
                   }
                   units seconds;
                   default 30;
```

```
                    description
                     "Sets the amount of time in seconds after
                      which if no data has been received from the
                      SSH/TLS server, a SSH/TLS-level message will
                      be sent to test the aliveness of the SSH/TLS
                      server.";
                  }
                  leaf max-attempts {
                    type uint8;
                    default 3;
                    description
                     "Sets the maximum number of sequential keep-
                      alive messages that can fail to obtain a
                      response from the SSH/TLS server before
                      assuming the SSH/TLS server is no longer
                      alive.";
                  }
                }
              }
            }
            case periodic-connection {
              container periodic {
                presence
                 "Indicates that a periodic connection is to be
                  maintained.";
                description
                 "Periodically connect to the NETCONF server.  The
                  NETCONF server should close the connection upon
                  completing planned activities.

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

```
                        series of periodic connections are determined.
                        The periodic connections occur at a whole
                        multiple interval from the anchor time.  For
                        example, for an anchor time is 15 minutes past
                        midnight and a period interval of 24 hours, then
                        a periodic connection will occur 15 minutes past
                        midnight everyday.";
                   }
                   leaf idle-timeout {
                     type uint16;
                     units "seconds";
                     default 120; // two minutes
                     description
                       "Specifies the maximum number of seconds that
                        a NETCONF session may remain idle. A NETCONF
                        session will be dropped if it is idle for an
                        interval longer than this number of seconds.
                        If set to zero, then the NETCONF client will
                        never drop a session because it is idle.";
                   }
                 }
               }
             }
           }
           container reconnect-strategy {
             description
              "The reconnection strategy directs how a NETCONF client
               reconnects to a NETCONF server, after discovering its
               connection to the server has dropped, even if due to a
               reboot.  The NETCONF client starts with the specified
               endpoint and tries to connect to it max-attempts times
               before trying the next endpoint in the list (round
               robin).";
             leaf start-with {
               type enumeration {
                 enum first-listed {
                   description
                     "Indicates that reconnections should start with
                      the first endpoint listed.";
                 }
                 enum last-connected {
                   description
                     "Indicates that reconnections should start with
                      the endpoint last connected to.  If no previous
                      connection has ever been established, then the
                      first endpoint configured is used.   NETCONF
                      clients SHOULD be able to remember the last
                      endpoint connected to across reboots.";
```

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

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

        leaf idle-timeout {
          type uint16;
          units "seconds";
          default 3600; // one hour
          description
            "Specifies the maximum number of seconds that a NETCONF
             session may remain idle. A NETCONF session will be
             dropped if it is idle for an interval longer than this
             number of seconds.  If set to zero, then the server
             will never drop a session because it is idle.  Sessions
             that have a notification subscription active are never
             dropped.";
        }

        list endpoint {
```

```
           key name;
           min-elements 1;
           description
             "List of endpoints to listen for NETCONF connections.";
           leaf name {
             type string;
             description
               "An arbitrary name for the NETCONF listen endpoint.";
           }
           choice transport {
             mandatory true;
             description
               "Selects between available transports.";
             case ssh {
               if-feature ssh-listen;
               container ssh {
                 description
                   "SSH-specific listening configuration for inbound
                    connections.";
                 leaf address {
                   type inet:ip-address;
                   description
                    "The IP address to listen on for incoming call-
                     home connections.  The NETCONF client will listen
                     on all configured interfaces if no value is
                     specified.  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 port {
                   type inet:port-number;
                   default 4334;
                   description
                    "The port number to listen on for call-home
                     connections.  The NETCONF client will listen
                     on the IANA-assigned well-known port for
                     'netconf-ch-ssh' (4334) if no value is
                     specified.";
                 }
                 uses ss:ssh-client-grouping;
               }
             }
             case tls {
               if-feature tls-listen;
               container tls {
                 description
                   "TLS-specific listening configuration for inbound
```

```
                    connections.";
                 leaf address {
                   type inet:ip-address;
                   description
                    "The IP address to listen on for incoming call-
                     home connections.  The NETCONF client will listen
                     on all configured interfaces if no value is
                     specified.  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 port {
                   type inet:port-number;
                   default 4335;
                   description
                    "The port number to listen on for call-home
                     connections.  The NETCONF client will listen
                     on the IANA-assigned well-known port for
                     'netconf-ch-tls' (4335) if no value is
                     specified.";
                 }
                 uses ts:tls-client-grouping {
                   refine "client-identity/auth-type" {
                     mandatory true;
                     description
                       "NETCONF/TLS clients MUST pass some
                        authentication credentials.";
                   }
                 }
               }
             }
           } // end transport
         } // end endpoint
       } // end listen

     } // end netconf-client
   }
   <CODE ENDS>
```

4. The NETCONF Server Model

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

This model supports both the SSH and TLS transport protocols, using the SSH server and TLS server groupings defined in

[I-D.ietf-netconf-ssh-client-server] and
[I-D.ietf-netconf-tls-client-server] respectively.

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

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

4.1. Tree Diagram

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

```
[Note: '\' line wrapping for formatting only]

module: ietf-netconf-server
  +--rw netconf-server
     +--rw listen! {listen}?
     |  +--rw idle-timeout?   uint16
     |  +--rw endpoint* [name]
     |     +--rw name         string
     |     +--rw (transport)
     |        +--:(ssh) {ssh-listen}?
     |        |  +--rw ssh
     |        |     +--rw address            inet:ip-address
     |        |     +--rw port?              inet:port-number
     |        |     +--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-keys-supported\
}?
     |        |     |        |        |  +--rw algorithm?
     |        |     |        |        |  |       asymmetric-key-encr\
yption-algorithm-ref
     |        |     |        |        |  +--rw public-key?
     |        |     |        |        |  |       binary
     |        |     |        |        |  +--rw private-key?
     |        |     |        |        |  |       union
     |        |     |        |        |  +---x generate-hidden-key
```

```
        |        |     |        |        | |  +---w input
        |        |     |        |        | |     +---w algorithm
        |        |     |        |        | |             asymmetric-ke\
   y-encryption-algorithm-ref
        |        |     |        |        |  +---x install-hidden-key
        |        |     |        |        |     +---w input
        |        |     |        |        |        +---w algorithm
        |        |     |        |        |        |       asymmetric-ke\
   y-encryption-algorithm-ref
        |        |     |        |        |        +---w public-key?
        |        |     |        |        |        |       binary
        |        |     |        |        |        +---w private-key?
        |        |     |        |        |                binary
        |        |     |        |       +--:(keystore)
        |        |     |        |                {keystore-supported}?
        |        |     |        |          +--rw reference?
        |        |     |        |                  ks:asymmetric-key-r\
   ef
        |        |     |       +--:(certificate)
        |        |     |          +--rw certificate
        |        |     |                  {sshcmn:ssh-x509-certs}?
        |        |     |             +--rw (local-or-keystore)
        |        |     |                +--:(local)
        |        |     |                |        {local-keys-supported\
   }?
        |        |     |                |  +--rw algorithm?
        |        |     |                |  |       asymmetric-key-encr\
   yption-algorithm-ref
        |        |     |                |  +--rw public-key?
        |        |     |                |  |       binary
        |        |     |                |  +--rw private-key?
        |        |     |                |  |       union
        |        |     |                |  +---x generate-hidden-key
        |        |     |                |  |  +---w input
        |        |     |                |  |     +---w algorithm
        |        |     |                |  |             asymmetric-ke\
   y-encryption-algorithm-ref
        |        |     |                |  +---x install-hidden-key
        |        |     |                |  |  +---w input
        |        |     |                |  |     +---w algorithm
        |        |     |                |  |     |       asymmetric-ke\
   y-encryption-algorithm-ref
        |        |     |                |  |     +---w public-key?
        |        |     |                |  |     |       binary
        |        |     |                |  |     +---w private-key?
        |        |     |                |  |             binary
        |        |     |                |  +--rw cert?
        |        |     |                |  |       end-entity-cert-cms
```

```
        |        |     |                 |  +---n certificate-expiration
        |        |     |                 |     +-- expiration-date
        |        |     |                 |             yang:date-and-ti\
   me
        |        |     |                 +--:(keystore)
        |        |     |                          {keystore-supported}?
        |        |     |                    +--rw reference?
        |        |     |                            ks:asymmetric-key-c\
   ertificate-ref
        |        |     +--rw client-cert-auth {sshcmn:ssh-x509-certs}?
        |        |     |  +--rw pinned-ca-certs?
        |        |     |  |       ta:pinned-certificates-ref
        |        |     |  |       {ta:x509-certificates}?
        |        |     |  +--rw pinned-client-certs?
        |        |     |          ta:pinned-certificates-ref
        |        |     |          {ta:x509-certificates}?
        |        |     +--rw transport-params
        |        |             {ssh-server-transport-params-config}?
        |        |        +--rw host-key
        |        |        |  +--rw host-key-alg*   identityref
        |        |        +--rw key-exchange
        |        |        |  +--rw key-exchange-alg*   identityref
        |        |        +--rw encryption
        |        |        |  +--rw encryption-alg*   identityref
        |        |        +--rw mac
        |        |           +--rw mac-alg*   identityref
        |        +--:(tls) {tls-listen}?
        |           +--rw tls
        |              +--rw address            inet:ip-address
        |              +--rw port?              inet:port-number
        |              +--rw server-identity
        |              |  +--rw (local-or-keystore)
        |              |     +--:(local) {local-keys-supported}?
        |              |     |  +--rw algorithm?
        |              |     |  |       asymmetric-key-encryption-algor\
   ithm-ref
        |              |     |  +--rw public-key?               binary
        |              |     |  +--rw private-key?              union
        |              |     |  +---x generate-hidden-key
        |              |     |  |  +---w input
        |              |     |  |     +---w algorithm
        |              |     |  |             asymmetric-key-encryption\
   -algorithm-ref
        |              |     |  +---x install-hidden-key
        |              |     |  |  +---w input
        |              |     |  |     +---w algorithm
        |              |     |  |     |       asymmetric-key-encryption\
   -algorithm-ref
```

```
        |              |     |  |     +---w public-key?    binary
        |              |     |  |     +---w private-key?   binary
        |              |     |  +--rw cert?
        |              |     |  |       end-entity-cert-cms
        |              |     |  +---n certificate-expiration
        |              |     |     +-- expiration-date
        |              |     |             yang:date-and-time
        |              |     +--:(keystore) {keystore-supported}?
        |              |        +--rw reference?
        |              |              ks:asymmetric-key-certificate-r\
   ef
        |              +--rw client-auth
        |              |  +--rw pinned-ca-certs?
        |              |  |       ta:pinned-certificates-ref
        |              |  |       {ta:x509-certificates}?
        |              |  +--rw pinned-client-certs?
        |              |  |       ta:pinned-certificates-ref
        |              |  |       {ta:x509-certificates}?
        |              |  +--rw cert-maps
        |              |     +--rw cert-to-name* [id]
        |              |        +--rw id             uint32
        |              |        +--rw fingerprint
        |              |        |       x509c2n:tls-fingerprint
        |              |        +--rw map-type       identityref
        |              |        +--rw name           string
        |              +--rw hello-params
        |                      {tls-server-hello-params-config}?
        |                 +--rw tls-versions
        |                 |  +--rw tls-version*   identityref
        |                 +--rw cipher-suites
        |                    +--rw cipher-suite*   identityref
        +--rw call-home! {call-home}?
           +--rw netconf-client* [name]
              +--rw name                  string
              +--rw endpoints
              |  +--rw endpoint* [name]
              |     +--rw name         string
              |     +--rw (transport)
              |        +--:(ssh) {ssh-call-home}?
              |        |  +--rw ssh
              |        |     +--rw address            inet:host
              |        |     +--rw port?              inet:port-number
              |        |     +--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-keys-sup\
ported}?
           |        |     |        |        |  +--rw algorithm?
           |        |     |        |        |  |       asymmetric-ke\
y-encryption-algorithm-ref
           |        |     |        |        |  +--rw public-key?
           |        |     |        |        |  |       binary
           |        |     |        |        |  +--rw private-key?
           |        |     |        |        |  |       union
           |        |     |        |        |  +---x generate-hidden\
-key
           |        |     |        |        |  |  +---w input
           |        |     |        |        |  |     +---w algorithm
           |        |     |        |        |  |             asymmet\
ric-key-encryption-algorithm-ref
           |        |     |        |        |  +---x install-hidden-\
key
           |        |     |        |        |     +---w input
           |        |     |        |        |        +---w algorithm
           |        |     |        |        |        |       asymmet\
ric-key-encryption-algorithm-ref
           |        |     |        |        |        +---w public-ke\
y?
           |        |     |        |        |        |       binary
           |        |     |        |        |        +---w private-k\
ey?
           |        |     |        |        |                binary
           |        |     |        |        +--:(keystore)
           |        |     |        |                 {keystore-suppo\
rted}?
           |        |     |        |           +--rw reference?
           |        |     |        |                   ks:asymmetric\
-key-ref
           |        |     |        +--:(certificate)
           |        |     |           +--rw certificate
           |        |     |                  {sshcmn:ssh-x509-certs\
}?
           |        |     |              +--rw (local-or-keystore)
           |        |     |                 +--:(local)
           |        |     |                 |        {local-keys-sup\
ported}?
           |        |     |                 |  +--rw algorithm?
           |        |     |                 |  |       asymmetric-ke\
y-encryption-algorithm-ref
           |        |     |                 |  +--rw public-key?
           |        |     |                 |  |       binary
```

```
              |        |     |                 |  +--rw private-key?
              |        |     |                 |  |       union
              |        |     |                 |  +---x generate-hidden\
   -key
              |        |     |                 |  |  +---w input
              |        |     |                 |  |     +---w algorithm
              |        |     |                 |  |             asymmet\
   ric-key-encryption-algorithm-ref
              |        |     |                 |  +---x install-hidden-\
   key
              |        |     |                 |  |  +---w input
              |        |     |                 |  |     +---w algorithm
              |        |     |                 |  |     |       asymmet\
   ric-key-encryption-algorithm-ref
              |        |     |                 |  |     +---w public-ke\
   y?
              |        |     |                 |  |     |       binary
              |        |     |                 |  |     +---w private-k\
   ey?
              |        |     |                 |  |             binary
              |        |     |                 |  +--rw cert?
              |        |     |                 |  |       end-entity-ce\
   rt-cms
              |        |     |                 |  +---n certificate-exp\
   iration
              |        |     |                 |     +-- expiration-date
              |        |     |                 |             yang:date-\
   and-time
              |        |     |                 +--:(keystore)
              |        |     |                          {keystore-suppo\
   rted}?
              |        |     |                    +--rw reference?
              |        |     |                            ks:asymmetric\
   -key-certificate-ref
              |        |     +--rw client-cert-auth
              |        |     |       {sshcmn:ssh-x509-certs}?
              |        |     |  +--rw pinned-ca-certs?
              |        |     |  |       ta:pinned-certificates-ref
              |        |     |  |       {ta:x509-certificates}?
              |        |     |  +--rw pinned-client-certs?
              |        |     |          ta:pinned-certificates-ref
              |        |     |          {ta:x509-certificates}?
              |        |     +--rw transport-params
              |        |             {ssh-server-transport-params-confi\
   g}?
              |        |        +--rw host-key
              |        |        |  +--rw host-key-alg*   identityref
              |        |        +--rw key-exchange
```

```
               |        |        |  +--rw key-exchange-alg*   identityref
               |        |        +--rw encryption
               |        |        |  +--rw encryption-alg*   identityref
               |        |        +--rw mac
               |        |           +--rw mac-alg*   identityref
               |        +--:(tls) {tls-call-home}?
               |           +--rw tls
               |              +--rw address            inet:host
               |              +--rw port?              inet:port-number
               |              +--rw server-identity
               |              |  +--rw (local-or-keystore)
               |              |     +--:(local) {local-keys-supported}?
               |              |     |  +--rw algorithm?
               |              |     |  |       asymmetric-key-encryption\
   -algorithm-ref
               |              |     |  +--rw public-key?
               |              |     |  |       binary
               |              |     |  +--rw private-key?
               |              |     |  |       union
               |              |     |  +---x generate-hidden-key
               |              |     |  |  +---w input
               |              |     |  |     +---w algorithm
               |              |     |  |             asymmetric-key-encr\
   yption-algorithm-ref
               |              |     |  +---x install-hidden-key
               |              |     |  |  +---w input
               |              |     |  |     +---w algorithm
               |              |     |  |     |       asymmetric-key-encr\
   yption-algorithm-ref
               |              |     |  |     +---w public-key?    binary
               |              |     |  |     +---w private-key?   binary
               |              |     |  +--rw cert?
               |              |     |  |       end-entity-cert-cms
               |              |     |  +---n certificate-expiration
               |              |     |     +-- expiration-date
               |              |     |             yang:date-and-time
               |              |     +--:(keystore) {keystore-supported}?
               |              |        +--rw reference?
               |              |                ks:asymmetric-key-certifi\
   cate-ref
               |              +--rw client-auth
               |              |  +--rw pinned-ca-certs?
               |              |  |       ta:pinned-certificates-ref
               |              |  |       {ta:x509-certificates}?
               |              |  +--rw pinned-client-certs?
               |              |  |       ta:pinned-certificates-ref
               |              |  |       {ta:x509-certificates}?
               |              |  +--rw cert-maps
```

```
             |               |     +--rw cert-to-name* [id]
             |               |        +--rw id             uint32
             |               |        +--rw fingerprint
             |               |        |       x509c2n:tls-fingerprint
             |               |        +--rw map-type       identityref
             |               |        +--rw name           string
             |               +--rw hello-params
             |                       {tls-server-hello-params-config}?
             |                  +--rw tls-versions
             |                  |  +--rw tls-version*   identityref
             |                  +--rw cipher-suites
             |                     +--rw cipher-suite*   identityref
             +--rw connection-type
             |  +--rw (connection-type)
             |     +--:(persistent-connection)
             |     |  +--rw persistent!
             |     |     +--rw keep-alives
             |     |        +--rw max-wait?       uint16
             |     |        +--rw max-attempts?   uint8
             |     +--:(periodic-connection)
             |        +--rw periodic!
             |           +--rw period?         uint16
             |           +--rw anchor-time?    yang:date-and-time
             |           +--rw idle-timeout?   uint16
             +--rw reconnect-strategy
                +--rw start-with?     enumeration
                +--rw max-attempts?   uint8
```

4.2. Example Usage

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

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

[Note: '\' line wrapping for formatting only]

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

```
                 <name>scooby-doo</name>
               </cert-to-name>
             </cert-maps>
           </client-auth>
         </tls>
       </endpoint>
     </listen>

     <!-- calling home to SSH and TLS based NETCONF clients -->
     <call-home>
       <netconf-client> <!-- SSH-based client -->
         <name>config-mgr</name>
         <endpoints>
           <endpoint>
             <name>east-data-center</name>
             <ssh>
               <address>east.config-mgr.example.com</address>
               <server-identity>
                 <host-key>
                   <name>deployment-specific-certificate</name>
                   <public-key>
                     <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\
   etf-crypto-types">ct:rsa2048</algorithm>
                     <private-key>base64encodedvalue==</private-key>
                     <public-key>base64encodedvalue==</public-key>
                   </public-key>
                 </host-key>
               </server-identity>
               <client-cert-auth>
                 <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
   inned-ca-certs>
                 <pinned-client-certs>explicitly-trusted-client-certs</\
   pinned-client-certs>
               </client-cert-auth>
             </ssh>
           </endpoint>
           <endpoint>
             <name>west-data-center</name>
             <ssh>
               <address>west.config-mgr.example.com</address>
               <server-identity>
                 <host-key>
                   <name>deployment-specific-certificate</name>
                   <public-key>
                     <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:i\
   etf-crypto-types">ct:rsa2048</algorithm>
                     <private-key>base64encodedvalue==</private-key>
                     <public-key>base64encodedvalue==</public-key>
```

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

```
                   <cert-to-name>
                     <id>2</id>
                     <fingerprint>B3:4F:A1:8C:54</fingerprint>
                     <map-type>x509c2n:specified</map-type>
                     <name>scooby-doo</name>
                   </cert-to-name>
                 </cert-maps>
               </client-auth>
             </tls>
           </endpoint>
           <endpoint>
             <name>west-data-center</name>
             <tls>
               <address>west.analytics.example.com</address>
               <server-identity>
                 <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
   crypto-types">ct:rsa2048</algorithm>
                 <private-key>base64encodedvalue==</private-key>
                 <public-key>base64encodedvalue==</public-key>
                 <cert>base64encodedvalue==</cert>
               </server-identity>
               <client-auth>
                 <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
   inned-ca-certs>
                 <pinned-client-certs>explicitly-trusted-client-certs</\
   pinned-client-certs>
                 <cert-maps>
                   <cert-to-name>
                     <id>1</id>
                     <fingerprint>11:0A:05:11:00</fingerprint>
                     <map-type>x509c2n:san-any</map-type>
                   </cert-to-name>
                   <cert-to-name>
                     <id>2</id>
                     <fingerprint>B3:4F:A1:8C:54</fingerprint>
                     <map-type>x509c2n:specified</map-type>
                     <name>scooby-doo</name>
                   </cert-to-name>
                 </cert-maps>
               </client-auth>
             </tls>
           </endpoint>
         </endpoints>
         <connection-type>
           <persistent>
             <keep-alives>
               <max-wait>30</max-wait>
               <max-attempts>3</max-attempts>
```

```
            </keep-alives>
          </persistent>
        </connection-type>
        <reconnect-strategy>
          <start-with>first-listed</start-with>
          <max-attempts>3</max-attempts>
        </reconnect-strategy>
      </netconf-client>
    </call-home>
  </netconf-server>
```


4.3. YANG Module

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

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

```
<CODE BEGINS> file "ietf-netconf-server@2018-10-22.yang"
module ietf-netconf-server {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-server";
  prefix "ncs";

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

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

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

  import ietf-ssh-server {
```

```
       prefix ss;
       revision-date 2018-10-22; // stable grouping definitions
       reference
         "RFC YYYY: YANG Groupings for SSH Clients and SSH Servers";
     }

     import ietf-tls-server {
       prefix ts;
       revision-date 2018-10-22; // stable grouping definitions
       reference
         "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
     }

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

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

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.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) 2017 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
```

```
       description
        "Initial version";
       reference
        "RFC XXXX: NETCONF Client and Server Models";
     }


     // Features

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

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

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

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

     feature ssh-call-home {
```

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

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


     // protocol accessible nodes

     container netconf-server {
       uses netconf-server-grouping;
       description
         "Top-level container for NETCONF server configuration.";
     }

     // reusable groupings

     grouping netconf-server-grouping {
       description
         "Top-level grouping for NETCONF server configuration.";
       container listen {
         if-feature listen;
         presence "Enables server to listen for TCP connections";
         description "Configures listen behavior";
         leaf idle-timeout {
           type uint16;
           units "seconds";
           default 3600; // one hour
           description
             "Specifies the maximum number of seconds that a NETCONF
              session may remain idle. A NETCONF session will be
              dropped if it is idle for an interval longer than this
              number of seconds.  If set to zero, then the server
              will never drop a session because it is idle.  Sessions
              that have a notification subscription active are never
              dropped.";
         }
```

```
         list endpoint {
           key name;
           min-elements 1;
           description
             "List of endpoints to listen for NETCONF connections.";
           leaf name {
             type string;
             description
               "An arbitrary name for the NETCONF listen endpoint.";
           }
           choice transport {
             mandatory true;
             description
               "Selects between available transports.";
             case ssh {
               if-feature ssh-listen;
               container ssh {
                 description
                   "SSH-specific listening configuration for inbound
                    connections.";
                 leaf address {
                   type inet:ip-address;
                   mandatory true;
                   description
                     "The IP address to listen on for incoming
                      connections.  The NETCONF server will listen
                      on all configured interfaces if no value is
                      specified.  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 port {
                   type inet:port-number;
                   default 830;
                   description
                    "The local port number to listen on.  If no value
                     is specified, the IANA-assigned port value for
                     'netconf-ssh' (830) is used.";
                 }
                 uses ss:ssh-server-grouping;
               }
             }
             case tls {
               if-feature tls-listen;
               container tls {
                 description
                   "TLS-specific listening configuration for inbound
```

```
                 connections.";
              leaf address {
                type inet:ip-address;
                mandatory true;
                description
                  "The IP address to listen on for incoming
                   connections.  The NETCONF server will listen
                   on all configured interfaces if no value is
                   specified.  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 port {
                type inet:port-number;
                default 6513;
                description
                 "The local port number to listen on.  If no value
                  is specified, the IANA-assigned port value for
                  'netconf-tls' (6513) is used.";
              }
              uses ts:tls-server-grouping {
                refine "client-auth" {
                  must 'pinned-ca-certs or pinned-client-certs';
                  description
                    "NETCONF/TLS servers MUST validate client
                     certiticates.";
                }
                augment "client-auth" {
                  description
                    "Augments in the cert-to-name structure.";
                  container cert-maps {
                    uses x509c2n:cert-to-name;
                    description
                     "The cert-maps container is used by a TLS-
                      based NETCONF server to map the NETCONF
                      client's presented X.509 certificate to a
                      NETCONF username.  If no matching and valid
                      cert-to-name list entry can be found, then
                      the NETCONF server MUST close the connection,
                      and MUST NOT accept NETCONF messages over
                      it.";
                    reference
                      "RFC WWWW: NETCONF over TLS, Section 7";
                  }
                }
              }
            }
```

```
              }
            }
          }
        }

        container call-home {
          if-feature call-home;
          presence "Enables server to initiate TCP connections";
          description "Configures call-home behavior";
          list netconf-client {
            key name;
            min-elements 1;
            description
              "List of NETCONF clients the NETCONF server is to
               initiate call-home connections to in parallel.";
            leaf name {
              type string;
              description
                "An arbitrary name for the remote NETCONF client.";
            }
            container endpoints {
              description
                "Container for the list of endpoints.";
              list endpoint {
                key name;
                min-elements 1;
                ordered-by user;
                description
                  "A non-empty user-ordered list of endpoints for this
                   NETCONF server to try to connect to in sequence.
                   Defining more than one enables high-availability.";
                leaf name {
                  type string;
                  description
                    "An arbitrary name for this endpoint.";
                }
                choice transport {
                  mandatory true;
                  description
                    "Selects between available transports.";
                  case ssh {
                    if-feature ssh-call-home;
                    container ssh {
                      description
                        "Specifies SSH-specific call-home transport
                         configuration.";
                      leaf address {
                        type inet:host;
```

```
                      mandatory true;
                      description
                       "The IP address or hostname of the endpoint.
                        If a domain name is configured, then the
                        DNS resolution should happen on each usage
                        attempt.  If the the DNS resolution results
                        in multiple IP addresses, the IP addresses
                        will be tried according to local preference
                        order until a connection has been established
                        or until all IP addresses have failed.";
                    }
                    leaf port {
                      type inet:port-number;
                      default 4334;
                      description
                       "The IP port for this endpoint. The NETCONF
                        server will use the IANA-assigned well-known
                        port for 'netconf-ch-ssh' (4334) if no value
                        is specified.";
                    }
                    uses ss:ssh-server-grouping;
                  }
                }
                case tls {
                  if-feature tls-call-home;
                  container tls {
                    description
                      "Specifies TLS-specific call-home transport
                       configuration.";
                    leaf address {
                      type inet:host;
                      mandatory true;
                      description
                       "The IP address or hostname of the endpoint.
                        If a domain name is configured, then the
                        DNS resolution should happen on each usage
                        attempt.  If the the DNS resolution results
                        in multiple IP addresses, the IP addresses
                        will be tried according to local preference
                        order until a connection has been established
                        or until all IP addresses have failed.";
                    }
                    leaf port {
                      type inet:port-number;
                      default 4335;
                      description
                       "The IP port for this endpoint.  The NETCONF
                        server will use the IANA-assigned well-known
```

```
                     port for 'netconf-ch-tls' (4335) if no value
                     is specified.";
                 }
                 uses ts:tls-server-grouping {
                   refine "client-auth" {
                    must 'pinned-ca-certs or pinned-client-certs';
                    description
                     "NETCONF/TLS servers MUST validate client
                      certificates.";
                   }
                   augment "client-auth" {
                     description
                       "Augments in the cert-to-name structure.";
                     container cert-maps {
                       uses x509c2n:cert-to-name;
                       description
                        "The cert-maps container is used by a
                         TLS-based NETCONF server to map the
                         NETCONF client's presented X.509
                         certificate to a NETCONF username.  If
                         no matching and valid cert-to-name list
                         entry can be found, then the NETCONF
                         server MUST close the connection, and
                         MUST NOT accept NETCONF messages over
                         it.";
                       reference
                         "RFC WWWW: NETCONF over TLS, Section 7";
                     }
                   }
                 }
               }
             } // end tls
           } // end choice
         } // end endpoint
       }
       container connection-type {
         description
          "Indicates the kind of connection to use.";
         choice connection-type {
           mandatory true;
           description
             "Selects between available connection types.";
           case persistent-connection {
             container persistent {
               presence
                "Indicates that a persistent connection is to be
                 maintained.";
               description
```

```
                 "Maintain a persistent connection to the NETCONF
                  client. If the connection goes down, immediately
                  start trying to reconnect to it, using the
                  reconnection strategy.

                  This connection type minimizes any NETCONF client
                  to NETCONF server data-transfer delay, albeit at
                  the expense of holding resources longer.";
                container keep-alives {
                  description
                    "Configures the keep-alive policy, to
                     proactively test the aliveness of the SSH/TLS
                     client.  An unresponsive SSH/TLS client will
                     be dropped after approximately max-attempts *
                     max-wait seconds.";
                  reference
                    "RFC 8071: NETCONF Call Home and RESTCONF
                               Call Home, Section 4.1, item S7";
                  leaf max-wait {
                    type uint16 {
                      range "1..max";
                    }
                    units seconds;
                    default 30;
                    description
                     "Sets the amount of time in seconds after
                      which if no data has been received from
                      the SSH/TLS client, a SSH/TLS-level message
                      will be sent to test the aliveness of the
                      SSH/TLS client.";
                  }
                  leaf max-attempts {
                    type uint8;
                    default 3;
                    description
                     "Sets the maximum number of sequential keep-
                     alive messages that can fail to obtain a
                     response from the SSH/TLS client before
                     assuming the SSH/TLS client is no longer
                     alive.";
                  }
                }
              }
            }

            case periodic-connection {
              container periodic {
                presence
```

```
                  "Indicates that a periodic connection is to be
                   maintained.";
                 description
                  "Periodically connect to the NETCONF client.  The
                   NETCONF client should close the underlying TLS
                   connection upon completing planned activities.

                   This connection type increases resource
                   utilization, albeit with increased delay in
                   NETCONF client to NETCONF client interactions.";
                 leaf period {
                   type uint16;
                   units "minutes";
                   default 60;
                   description
                     "Duration of time between periodic connections.";
                 }
                 leaf anchor-time {
                   type yang:date-and-time {
                     // constrained to minute-level granularity
                     pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}'
                             + '(Z|[\+\-]\d{2}:\d{2})';
                   }
                   description
                     "Designates a timestamp before or after which a
                      series of periodic connections are determined.
                      The periodic connections occur at a whole
                      multiple interval from the anchor time.  For
                      example, for an anchor time is 15 minutes past
                      midnight and a period interval of 24 hours, then
                      a periodic connection will occur 15 minutes past
                      midnight everyday.";
                 }
                 leaf idle-timeout {
                   type uint16;
                   units "seconds";
                   default 120; // two minutes
                   description
                     "Specifies the maximum number of seconds that
                      a NETCONF session may remain idle. A NETCONF
                      session will be dropped if it is idle for an
                      interval longer than this number of seconds.
                      If set to zero, then the server will never
                      drop a session because it is idle.";
                 }
               }
             }
           }
```

```
           }
           container 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).";
             }
```

```
           }
         }
       }
     }
   }
```

   <CODE ENDS>

5. Design Considerations

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

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

5.1. Support all NETCONF transports

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

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

5.2. Enable each transport to select which keys to use

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

5.3. Support authenticating NETCONF clients certificates

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

5.4. Support mapping authenticated NETCONF client certificates to usernames

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

5.5. Support both listening for connections and call home

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

5.6. For Call Home connections

The following objectives only pertain to call home connections.

5.6.1. Support more than one NETCONF client

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

5.6.2. Support NETCONF clients having more than one endpoint

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

5.6.3. Support a reconnection strategy

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

5.6.4. Support both persistent and periodic connections

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

5.6.5. Reconnection strategy for periodic connections

The reconnection strategy should apply to both persistent and periodic connections. How it applies to periodic connections becomes clear when considering that a periodic "connection" is a logical connection to a single server. That is, the periods of unconnectedness are intentional as opposed to due to external reasons. A periodic "connection" should always reconnect to the same server until it is no longer able to, at which time the reconnection strategy guides how to connect to another server.

5.6.6. Keep-alives for persistent connections

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

5.6.7. Customizations for periodic connections

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

6. Security Considerations

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

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

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

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

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

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

NONE

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

NONE

7. IANA Considerations

7.1. The IETF XML Registry

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

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

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

7.2. The YANG Module Names Registry

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

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

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

8. References

8.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-06 (work in progress), September 2018.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", draft-ietf-netconf-ssh-client-server-07 (work in progress), September 2018.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-07 (work in progress), September 2018.

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

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

8.2. Informative References

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

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

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

Appendix A. Change Log

A.1. 00 to 01

- Renamed "keychain" to "keystore".

A.2. 01 to 02

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

A.3. 02 to 03

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

A.4. 03 to 04

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

A.5. 04 to 05

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

A.6. 05 to 06

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

A.7. 06 to 07

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

A.8. 07 to 08

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

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.

Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

A. Gonzalez Prieto
Microsoft
E. Nilsen-Nygaard
A. Tripathy
Cisco Systems
October 23, 2018

Dynamic subscription to YANG Events and Datastores over NETCONF
draft-ietf-netconf-netconf-event-notifications-14

Abstract

This document provides a NETCONF binding to the dynamic subscription capability of both subscribed notifications and YANG push.

RFC Editor note: please replace the four references to pre-RFC normative drafts with the actual assigned RFC numbers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. Compatibility with RFC-5277's create-subscription 3
4. Mandatory XML, event stream and datastore support 3
5. NETCONF connectivity and the Dynamic Subscriptions 4
6. Notification Messages 4
7. Dynamic Subscriptions and RPC Error Responses 4
8. Security Considerations 5
9. Acknowledgments 6
10. Notes to the RFC Editor 6
11. References 6
11.1. Normative References 6
11.2. Informative References 7
Appendix A. Examples 7
A.1. Event Stream Discovery 7
A.2. Dynamic Subscriptions 8
A.3. Subscription State Notifications 12
A.4. Filter Examples 14
Appendix B. Changes between revisions 15
B.1. v13 to v14 15
B.2. v11 to v13 16
B.3. v10 to v11 16
B.4. v09 to v10 16
B.5. v08 to v09 16
B.6. v07 to v08 16
B.7. v06 to v07 16
B.8. v05 to v06 16
B.9. v03 to v04 16
B.10. v01 to v03 17
B.11. v00 to v01 17
Authors' Addresses 17

1. Introduction

This document provides a binding for events streamed over the NETCONF protocol [RFC6241] for dynamic subscriptions as defined in [I-D.draft-ietf-netconf-subscribed-notifications]. In addition, as [I-D.ietf-netconf-yang-push] is itself built upon [I-D.draft-ietf-netconf-subscribed-notifications], this document enables a NETCONF client to request via a dynamic subscription and receive updates from a YANG datastore located on a NETCONF server.

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.

The following terms are defined in [I-D.draft-ietf-netconf-subscribed-notifications]: dynamic subscription, event stream, notification message, publisher, receiver, subscriber, subscription. No additional terms are defined.

3. Compatibility with RFC-5277's create-subscription

A publisher is allowed to concurrently support dynamic subscription RPCs of [I-D.draft-ietf-netconf-subscribed-notifications] at the same time as [RFC5277]'s "create-subscription" RPC. However a single NETCONF transport session cannot support both this specification and a subscription established by [RFC5277]'s "create-subscription" RPC. To protect against any attempts to use a single NETCONF transport session in this way:

- o A solution must reply with the [RFC6241] error "operation-not-supported" if a "create-subscription" RPC is received on a NETCONF session where an [I-D.draft-ietf-netconf-subscribed-notifications] established subscription exists.
- o A solution must reply with the [RFC6241] error "operation-not-supported" if an "establish-subscription" request has been received on a NETCONF session where the "create-subscription" RPC has successfully [RFC5277] created a subscription.

If a publisher supports this specification but not subscriptions via [RFC5277], the publisher MUST NOT advertise "urn:ietf:params:netconf:capability:notification:1.0".

4. Mandatory XML, event stream and datastore support

The "encode-xml" feature of [I-D.draft-ietf-netconf-subscribed-notifications] MUST be supported. This indicates that XML is a valid encoding for RPCs, state change notifications, and subscribed content.

A NETCONF publisher supporting event stream subscription via [I-D.draft-ietf-netconf-subscribed-notifications] MUST support the "NETCONF" event stream identified in that document.

5. NETCONF connectivity and the Dynamic Subscriptions

For a dynamic subscription, if the NETCONF session involved with the "establish-subscription" terminates the subscription MUST be terminated.

For a dynamic subscription, any "modify-subscription", "delete-subscription", or "resynch-subscription" RPCs MUST be sent using the same NETCONF session upon which the referenced subscription was established.

6. Notification Messages

Notification messages transported over the NETCONF protocol MUST be encoded in a <notification> message as defined within [RFC5277], Section 4. And per [RFC5277]'s "eventTime" object definition, the "eventTime" MUST be populated with the event occurrence time.

For dynamic subscriptions, all notification messages MUST use the NETCONF transport session used by the "establish-subscription" RPC.

7. Dynamic Subscriptions and RPC Error Responses

Management of dynamic subscriptions occurs via RPCs as defined in [I-D.ietf-netconf-yang-push] and [I-D.draft-ietf-netconf-subscribed-notifications]. When an RPC error occurs, the NETCONF RPC reply MUST include an "rpc-error" element per [RFC6241] with the error information populated as follows:

- o an "error-type" node of "application".
- o an "error-tag" node of "operation-failed".
- o an "error-severity" of "error" (this MAY but does not have to be included).
- o an "error-app-tag" node with the value being a string that corresponds to an identity associated with the error, as defined in [I-D.draft-ietf-netconf-subscribed-notifications] section 2.4.6 for general subscriptions, and [I-D.ietf-netconf-yang-push] Appendix A.1, for datastore subscriptions. The specific identity to use depends on the RPC for which the error occurred. Each error identity will be inserted as the "error-app-tag" following the form <modulename>:<identityname>. An example of such as valid encoding would be "ietf-subscribed-notifications:no-such-subscription". Viable errors for different RPCs are as follows:

RPC	use base identity
establish-subscription	establish-subscription-error
modify-subscription	modify-subscription-error
delete-subscription	delete-subscription-error
kill-subscription	kill-subscription-error
resynch-subscription	resynch-subscription-error

o In case of error responses to an "establish-subscription" or "modify-subscription" request there is the option of including an "error-info" node. This node may contain XML-encoded data with hints for parameter settings that might lead to successful RPC requests in the future. Following are the yang-data structures from [I-D.draft-ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push] which may be returned:

establish-subscription	returns hints in yang-data structure
target: event stream	establish-subscription-stream-error-info
target: datastore	establish-subscription-datastore-error-info

modify-subscription	returns hints in yang-data structure
target: event stream	modify-subscription-stream-error-info
target: datastore	modify-subscription-datastore-error-info

The yang-data included within "error-info" SHOULD NOT include the optional leaf "error-reason", as such a leaf would be redundant with information that is already placed within the "error-app-tag".

In case of an rpc error resulting from a "delete-subscription", "kill-subscription", or "resynch-subscription" request, no "error-info" needs to be included, as the "subscription-id" is the only RPC input parameter and no hints regarding this RPC input parameters need to be provided.

8. Security Considerations

If a malicious or buggy NETCONF subscriber sends a number of establish-subscription requests, then these subscriptions accumulate and may use up system resources. In such a situation, subscriptions MAY be terminated by terminating the underlying NETCONF session. The publisher MAY also suspend or terminate a subset of the active subscriptions on that NETCONF session.

9. Acknowledgments

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from: Andy Bierman, Yan Gang, Sharon Chisholm, Hector Trevino, Peipei Guo, Susan Hares, Tim Jenkins, Balazs Lengyel, Martin Bjorklund, Mahesh Jethanandani, Kent Watsen, and Guangying Zheng.

10. Notes to the RFC Editor

This section can be removed by the RFC editor after the requests have been performed.

RFC 6241 need to be updated. RFC-6241 refers to RFC-5277 which says that a notification message can only be sent after a successful "create-subscription". This text must be modified to also allow notification messages be sent after a successful "establish-subscription".

11. References

11.1. Normative References

[I-D.draft-ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Gonzalez Prieto, A., Tripathy, A., and E. Nilsen-Nygaard, "Customized Subscriptions to a Publisher's Event Streams", September 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-subscribed-notifications/>.

[I-D.ietf-netconf-yang-push]
Clemm, Alexander., Voit, Eric., Gonzalez Prieto, Alberto., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", September 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

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

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

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

11.2. Informative References

[RFC8347] Liu, X., Ed., Kyparlis, A., Parikh, R., Lindem, A., and M. Zhang, "A YANG Data Model for the Virtual Router Redundancy Protocol (VRRP)", RFC 8347, DOI 10.17487/RFC8347, March 2018, <https://www.rfc-editor.org/info/rfc8347>.

[XPATH] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

Appendix A. Examples

This section is non-normative.

A.1. Event Stream Discovery

As defined in [I-D.draft-ietf-netconf-subscribed-notifications] an event stream exposes a continuous set of events available for subscription. A NETCONF client can retrieve the list of available event streams from a NETCONF publisher using the "get" operation against the top-level container "/streams" defined in [I-D.draft-ietf-netconf-subscribed-notifications] Section 3.1.

The following example illustrates the retrieval of the list of available event streams:

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <streams
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"/>
    </filter>
  </get>
</rpc>
```

Figure 1: Get streams request

After such a request, the NETCONF publisher returns a list of event streams available, as well as additional information which might exist in the container.

A.2. Dynamic Subscriptions

A.2.1. Establishing Dynamic Subscriptions

The following figure shows two successful "establish-subscription" RPC requests as per [I-D.draft-ietf-netconf-subscribed-notifications]. The first request is given a subscription "id" of 22, the second, an "id" of 23.

```
      +------------+                 +-----------+
      | Subscriber |                 | Publisher |
      +------------+                 +-----------+
            |                              |
            |    Capability Exchange       |
            |<---------------------------->|
            |                              |
            |                              |
            |    establish-subscription    |
            |----------------------------->|  (a)
            | RPC Reply: OK, id = 22       |
            |<-----------------------------|  (b)
            |                              |
            | notification message (for 22)|
            |<-----------------------------|
            |                              |
            |                              |
            |    establish-subscription    |
            |----------------------------->|
            | notification message (for 22)|
            |<-----------------------------|
            | RPC Reply: OK, id = 23       |
            |<-----------------------------|
            |                              |
            |                              |
            | notification message (for 22)|
            |<-----------------------------|
            | notification message (for 23)|
            |<-----------------------------|
            |                              |
```

Figure 2: Multiple subscriptions over a NETCONF session

To provide examples of the information being transported, example messages for interactions (a) and (b) in Figure 2 are detailed below:

```
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <stream>NETCONF</stream>
    <stream-xpath-filter xmlns:ds="http://example.com/events">
      /ds:foo/
    </stream-xpath-filter>
    <dscp>10</dscp>
  </establish-subscription>
</rpc>
```

Figure 3: establish-subscription request (a)

As NETCONF publisher was able to fully satisfy the request (a), the publisher sends the subscription "id" of the accepted subscription within message (b):

```
<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <id
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    22
  </id>
</rpc-reply>
```

Figure 4: establish-subscription success (b)

If the NETCONF publisher had not been able to fully satisfy the request, or subscriber has no authorization to establish the subscription, the publisher would have sent an RPC error response. For instance, if the "dscp" value of 10 asserted by the subscriber in Figure 3 proved unacceptable, the publisher may have returned:

```
<rpc-reply message-id="102"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
   <error-type>application</error-type>
   <error-tag>operation-failed</error-tag>
   <error-severity>error</error-severity>
   <error-app-tag>
     ietf-subscribed-notifications:dscp-unavailable
   </error-app-tag>
  </rpc-error>
</rpc-reply>
```

Figure 5: an unsuccessful establish subscription

The subscriber can use this information in future attempts to establish a subscription.

A.2.2. Modifying Dynamic Subscriptions

An existing subscription may be modified. The following exchange shows a negotiation of such a modification via several exchanges between a subscriber and a publisher. This negotiation consists of a failed RPC modification request/response, followed by a successful one.

```
+------------+                 +-----------+
| Subscriber |                 | Publisher |
+------------+                 +-----------+
      |                              |
      | notification message (for 23)|
      |<-----------------------------|
      |                              |
      | modify-subscription (id = 23)|
      |----------------------------->|  (c)
      | RPC error (with hint)        |
      |<-----------------------------|  (d)
      |                              |
      | modify-subscription (id = 23)|
      |----------------------------->|
      | RPC Reply: OK                |
      |<-----------------------------|
      |                              |
      | notification message (for 23)|
      |<-----------------------------|
      |                              |
```

Figure 6: Interaction model for successful subscription modification

If the subscription being modified in Figure 6 is a datastore subscription as per [I-D.ietf-netconf-yang-push], the modification request made in (c) may look like that shown in Figure 7. As can be seen, the modifications being attempted are the application of a new XPath filter as well as the setting of a new periodic time interval.

```
<rpc message-id="303"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
       xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
       xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>23</id>
    <yp:datastore-xpath-filter xmlns:ds="http://example.com/datastore">
        /ds:foo/ds:bar
    </yp:datastore-xpath-filter>
    <yp:periodic>
      <yp:period>500</yp:period>
    </yp:periodic>
  </modify-subscription>
</rpc>
```

Figure 7: Subscription modification request (c)

If the NETCONF publisher can satisfy both changes, the publisher sends a positive result for the RPC. If the NETCONF publisher cannot satisfy either of the proposed changes, the publisher sends an RPC error response (d). The following is an example RPC error response for (d) which includes a hint. This hint is an alternative time period value which might have resulted in a successful modification:

```
<rpc-reply message-id="303"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-app-tag>
        ietf-yang-push:period-unsupported
    </error-app-tag>
    <error-info>
      <modify-subscription-datastore-error-info
          xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <period-hint>
            3000
        </period-hint>
      </modify-subscription-datastore-error-info>
    </error-info>
  </rpc-error>
</rpc-reply>
```

Figure 8: Modify subscription failure with hint (d)

A.2.3. Deleting Dynamic Subscriptions

The following demonstrates deleting a subscription. This subscription may have been to either a stream or a datastore.

```
<rpc message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <id>22</id>
  </delete-subscription>
</rpc>
```

Figure 9: Delete subscription

If the NETCONF publisher can satisfy the request, the publisher replies with success to the RPC request.

If the NETCONF publisher cannot satisfy the request, the publisher sends an error-rpc element indicating the modification didn't work. Figure 10 shows a valid response for existing valid subscription "id", but that subscription "id" was created on a different NETCONF transport session:

```
<rpc-reply message-id="103"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-app-tag>
        ietf-subscribed-notifications:no-such-subscription
    </error-app-tag>
  </rpc-error>
</rpc-reply>
```

Figure 10: Unsuccessful delete subscription

A.3. Subscription State Notifications

A publisher will send subscription state notifications for dynamic subscriptions according to the definitions within [I-D.draft-ietf-netconf-subscribed-notifications].

A.3.1. subscription-modified

As per Section 2.7.2 of [I-D.draft-ietf-netconf-subscribed-notifications], a "subscription-modified" might be sent over NETCONF if the definition of a configured filter changes. A subscription state notification encoded in XML would look like:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-modified
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <id>39</id>
    <stream-xpath-filter xmlns:ex="http://example.com/events">
      /ex:foo
    </stream-xpath-filter>
    <stream>NETCONF</stream>
  </subscription-modified>
</notification>
```

Figure 11: subscription-modified subscription state notification

A.3.2. subscription-resumed, and replay-complete

A "subscription-resumed" would look like:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-resumed
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <id>39</id>
  </subscription-resumed>
</notification>
```

Figure 12: subscription-resumed notification in XML

The "replay-complete" is virtually identical, with "subscription-resumed" simply being replaced by "replay-complete".

A.3.3. subscription-terminated and subscription-suspended

A "subscription-terminated" would look like:

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <subscription-terminated
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <id>39</id>
    <reason>
       suspension-timeout
    </reason>
  </subscription-terminated>
</notification>
```

Figure 13: subscription-terminated subscription state notification

The "subscription-suspended" is virtually identical, with "subscription-terminated" simply being replaced by "subscription-suspended".

A.4. Filter Examples

This section provides examples which illustrate both XPath and subtree methods of filtering event record contents. The examples are based on the YANG notification "vrrp-protocol-error-event" as defined per the ietf-vrrp.yang model within [RFC8347]. Event records based on this specification which are generated by the publisher might appear as:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-09-14T08:22:33.44Z</eventTime>
  <vrrp-protocol-error-event
       xmlns="urn:ietf:params:xml:ns:yang:ietf-vrrp">
     <protocol-error-reason>checksum-error</protocol-error-reason>
  </vrrp-protocol-error-event>
</notification>
```

Figure 14: RFC 8347 (VRRP) - Example Notification

Suppose a subscriber wanted to establish a subscription which only passes instances of event records where there is a "checksum-error" as part of a VRRP protocol event. Also assume the publisher places such event records into the NETCONF stream. To get a continuous series of matching event records, the subscriber might request the application of an XPath filter against the NETCONF stream. An "establish-subscription" RPC to meet this objective might be:

```
<rpc message-id="601" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
    xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <stream>NETCONF</stream>
    <stream-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-vrrp">
      /vrrp-protocol-error-event[
         vrrp:protocol-error-reason="vrrp:checksum-error"]
    </stream-xpath-filter>
  </establish-subscription>
</rpc>
```

Figure 15: Establishing a subscription error reason via XPath

For more examples of XPath filters, see [XPATH].

Suppose the "establish-subscription" in Figure 15 was accepted. And suppose later a subscriber decided they wanted to broaden this subscription cover to all VRRP protocol events (i.e., not just those with a "checksum error"). The subscriber might attempt to modify the subscription in a way which replaces the XPath filter with a subtree filter which sends all VRRP protocol events to a subscriber. Such a "modify-subscription" RPC might look like:

```
<rpc message-id="602" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <modify-subscription
     xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <id>99</id>
    <stream-subtree-filter>
     <vrrp-protocol-error-event
            xmlns="urn:ietf:params:xml:ns:yang:ietf-vrrp"/>
    </stream-subtree-filter>
  </modify-subscription>
</rpc>
```

Figure 16

For more examples of subtree filters, see [RFC6241], section 6.4.

Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

B.1. v13 to v14

- o Title change.

B.2. v11 to v13

- o Subscription identifier renamed to id.
- o Appendix A.4 for filter examples
- o for v13, Tweak of example to /foo/bar

B.3. v10 to v11

- o Configured removed.

B.4. v09 to v10

- o Tweaks to examples and text.
- o Downshifted state names.
- o Removed address from examples.

B.5. v08 to v09

- o Tweaks based on Kent's comments.
- o Updated examples in Appendix A. And updates to some object names based on changes in the subscribed-notifications draft.
- o Added a YANG model for the NETCONF identity.

B.6. v07 to v08

- o Tweaks and clarification on :interleave.

B.7. v06 to v07

- o XML encoding and operational datastore mandatory.
- o Error mechanisms and examples updated.

B.8. v05 to v06

- o Moved examples to appendices
- o All examples rewritten based on namespace learnings
- o Normative text consolidated in front
- o Removed all mention of JSON
- o Call home process detailed
- o Note: this is a major revision attempting to cover those comments received from two week review.

B.9. v03 to v04

- o Added additional detail to "configured subscriptions"
- o Added interleave capability
- o Adjusted terminology to that in draft-ietf-netconf-subscribed-notifications

o Corrected namespaces in examples

B.10. v01 to v03

o Text simplifications throughout
o v02 had no meaningful changes

B.11. v00 to v01

o Added Call Home in solution for configured subscriptions.
o Clarified support for multiple subscription on a single session. No need to support multiple create-subscription.
o Added mapping between terminology in yang-push and [RFC6241] (the one followed in this document).
o Editorial improvements.

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Alberto Gonzalez Prieto
Microsoft

Email: alberto.gonzalez@microsoft.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Ambika Prasad Tripathy
Cisco Systems

Email: ambtripa@cisco.com

NETCONF Working Group K. Watsen
Internet-Draft Juniper Networks
Intended status: Standards Track October 22, 2018
Expires: April 25, 2019

RESTCONF Client and Server Models
draft-ietf-netconf-restconf-client-server-08

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

o I-D.ietf-netconf-keystore

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

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

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

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

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

o "2018-10-22" --> the publication date of this draft

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

o Appendix A. Change Log

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Table of Contents

1. Introduction 3
 1.1. Terminology 3
2. The RESTCONF Client Model 3
 2.1. Tree Diagram 4
 2.2. Example Usage 7
 2.3. YANG Module 9
3. The RESTCONF Server Model 18
 3.1. Tree Diagram 18
 3.2. Example Usage 21
 3.3. YANG Module 24
4. Security Considerations 34
5. IANA Considerations 35
 5.1. The IETF XML Registry 35
 5.2. The YANG Module Names Registry 35

6. References 36
6.1. Normative References 36
6.2. Informative References 37
Appendix A. Change Log 38
A.1. 00 to 01 38
A.2. 01 to 02 38
A.3. 02 to 03 38
A.4. 03 to 04 38
A.5. 04 to 05 38
A.6. 05 to 06 39
A.7. 06 to 07 39
A.8. 07 to 08 39
Acknowledgements 39
Author's Address 39

1. Introduction

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

1.1. Terminology

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

2. The RESTCONF Client Model

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

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

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

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

2.1. Tree Diagram

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

```
[Note: '\' line wrapping for formatting only]

module: ietf-restconf-client
  +--rw restconf-client
     +--rw initiate! {initiate}?
     |  +--rw restconf-server* [name]
     |     +--rw name         string
     |     +--rw endpoints
     |        +--rw endpoint* [name]
     |           +--rw name                   string
     |           +--rw (transport)
     |           |  +--:(tls) {tls-initiate}?
     |           |     +--rw tls
     |           |        +--rw address            inet:host
     |           |        +--rw port?              inet:port-number
     |           |        +--rw client-identity
     |           |        |  +--rw (auth-type)
     |           |        |     +--:(certificate)
     |           |        |        +--rw certificate
     |           |        |           +--rw (local-or-keystore)
     |           |        |              +--:(local)
     |           |        |              |        {local-keys-suppor\
ted}?
     |           |        |              |  +--rw algorithm?
     |           |        |              |  |       asymmetric-key-e\
ncryption-algorithm-ref
     |           |        |              |  +--rw public-key?
     |           |        |              |  |       binary
     |           |        |              |  +--rw private-key?
     |           |        |              |  |       union
     |           |        |              |  +---x generate-hidden-key
     |           |        |              |  |  +---w input
     |           |        |              |  |     +---w algorithm
     |           |        |              |  |             asymmetric\
-key-encryption-algorithm-ref
     |           |        |              |  +---x install-hidden-key
     |           |        |              |  |  +---w input
     |           |        |              |  |     +---w algorithm
     |           |        |              |  |     |       asymmetric\
-key-encryption-algorithm-ref
```

```
        |           |        |              |  |     +---w public-key?
        |           |        |              |  |     |       binary
        |           |        |              |  |     +---w private-key?
        |           |        |              |  |             binary
        |           |        |              |  +--rw cert?
        |           |        |              |  |        end-entity-cert-\
   cms
        |           |        |              |  +---n certificate-expira\
   tion
        |           |        |              |     +-- expiration-date
        |           |        |              |                yang:date-and\
   -time
        |           |        |              +--:(keystore)
        |           |        |                       {keystore-supporte\
   d}?
        |           |        |                 +--rw reference?
        |           |        |                         ks:asymmetric-ke\
   y-certificate-ref
        |           |        +--rw server-auth
        |           |        |  +--rw pinned-ca-certs?
        |           |        |  |       ta:pinned-certificates-ref
        |           |        |  |       {ta:x509-certificates}?
        |           |        |  +--rw pinned-server-certs?
        |           |        |          ta:pinned-certificates-ref
        |           |        |          {ta:x509-certificates}?
        |           |        +--rw hello-params
        |           |                {tls-client-hello-params-config}?
        |           |           +--rw tls-versions
        |           |           |  +--rw tls-version*   identityref
        |           |           +--rw cipher-suites
        |           |              +--rw cipher-suite*   identityref
        |           +--rw connection-type
        |           |  +--rw (connection-type)
        |           |     +--:(persistent-connection)
        |           |     |  +--rw persistent!
        |           |     |     +--rw keep-alives
        |           |     |        +--rw max-wait?       uint16
        |           |     |        +--rw max-attempts?   uint8
        |           |     +--:(periodic-connection)
        |           |        +--rw periodic!
        |           |           +--rw period?         uint16
        |           |           +--rw anchor-time?    yang:date-and-time
        |           |           +--rw idle-timeout?   uint16
        |           +--rw reconnect-strategy
        |              +--rw start-with?     enumeration
        |              +--rw max-attempts?   uint8
        +--rw listen! {listen}?
           +--rw idle-timeout?   uint16
```

```
           +--rw endpoint* [name]
              +--rw name         string
              +--rw (transport)
                 +--:(tls) {tls-listen}?
                    +--rw tls
                       +--rw address?           inet:ip-address
                       +--rw port?              inet:port-number
                       +--rw client-identity
                       |  +--rw (auth-type)
                       |     +--:(certificate)
                       |        +--rw certificate
                       |           +--rw (local-or-keystore)
                       |              +--:(local) {local-keys-supported\
   }?
                       |              |  +--rw algorithm?
                       |              |  |       asymmetric-key-encrypt\
   ion-algorithm-ref
                       |              |  +--rw public-key?
                       |              |  |       binary
                       |              |  +--rw private-key?
                       |              |  |       union
                       |              |  +---x generate-hidden-key
                       |              |  |  +---w input
                       |              |  |     +---w algorithm
                       |              |  |             asymmetric-key-e\
   ncryption-algorithm-ref
                       |              |  +---x install-hidden-key
                       |              |  |  +---w input
                       |              |  |     +---w algorithm
                       |              |  |     |       asymmetric-key-e\
   ncryption-algorithm-ref
                       |              |  |     +---w public-key?    bin\
   ary
                       |              |  |     +---w private-key?   bin\
   ary
                       |              |  +--rw cert?
                       |              |  |       end-entity-cert-cms
                       |              |  +---n certificate-expiration
                       |              |     +-- expiration-date
                       |              |             yang:date-and-time
                       |              +--:(keystore) {keystore-supporte\
   d}?
                       |                 +--rw reference?
                       |                         ks:asymmetric-key-cert\
   ificate-ref
                       +--rw server-auth
                       |  +--rw pinned-ca-certs?
                       |  |       ta:pinned-certificates-ref
```

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

2.2. Example Usage

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

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

[Note: '\' line wrapping for formatting only]


```
<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>
          <tls>
            <address>corp-fw1.example.com</address>
            <client-identity>
              <certificate>
                <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
f-crypto-types">ct:rsa2048</algorithm>
                <private-key>base64encodedvalue==</private-key>
                <public-key>base64encodedvalue==</public-key>
                <cert>base64encodedvalue==</cert>
              </certificate>
            </client-identity>
            <server-auth>
              <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
inned-ca-certs>
              <pinned-server-certs>explicitly-trusted-server-certs</\
pinned-server-certs>
```

```
              </server-auth>
            </tls>
            <connection-type>
              <persistent/>
            </connection-type>
          </endpoint>
          <endpoint>
            <name>corp-fw2.example.com</name>
            <tls>
              <address>corp-fw2.example.com</address>
              <client-identity>
                <certificate>
                  <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:iet\
   f-crypto-types">ct:rsa2048</algorithm>
                  <private-key>base64encodedvalue==</private-key>
                  <public-key>base64encodedvalue==</public-key>
                  <cert>base64encodedvalue==</cert>
                </certificate>
              </client-identity>
              <server-auth>
                <pinned-ca-certs>explicitly-trusted-server-ca-certs</p\
   inned-ca-certs>
                <pinned-server-certs>explicitly-trusted-server-certs</\
   pinned-server-certs>
              </server-auth>
            </tls>
            <connection-type>
              <persistent/>
            </connection-type>
          </endpoint>
        </endpoints>
      </restconf-server>
    </initiate>

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

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

2.3. YANG Module

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

```
<CODE BEGINS> file "ietf-restconf-client@2018-10-22.yang"
module ietf-restconf-client {
  yang-version 1.1;

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

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

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

  import ietf-tls-client {
    prefix ts;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
  }

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

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

```
       WG List:  <mailto:restconf@ietf.org>

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>

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


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

       Copyright (c) 2017 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: RESTCONF Client and Server Models";
     }

     // Features

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

     feature tls-initiate {
       if-feature initiate;
       description
        "The 'tls-initiate' feature indicates that the RESTCONF client
         supports initiating TLS connections to RESTCONF servers. This
         feature exists as TLS might not be a mandatory to implement
```

```
         transport in the future.";
       reference
        "RFC 8040: RESTCONF Protocol";
     }

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

     feature tls-listen {
       if-feature listen;
       description
        "The 'tls-listen' feature indicates that the RESTCONF client
         supports opening a port to listen for incoming RESTCONF
         server call-home TLS connections.  This feature exists as
         TLS might not be a mandatory to implement transport in the
         future.";
       reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
     }

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

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

       container initiate {
         if-feature initiate;
         presence "Enables client to initiate TCP connections";
         description
           "Configures client initiating underlying TCP connections.";
         list restconf-server {
           key name;
           min-elements 1;
           description
             "List of RESTCONF servers the RESTCONF client is to
              initiate connections to in parallel.";
           leaf name {
             type string;
```

```
             description
               "An arbitrary name for the RESTCONF server.";
           }
           container endpoints {
             description
               "Container for the list of endpoints.";
             list endpoint {
               key name;
               min-elements 1;
               ordered-by user;
               description
                 "A non-empty user-ordered list of endpoints for this
                  RESTCONF client to try to connect to in sequence.
                  Defining more than one enables high-availability.";
               leaf name {
                 type string;
                 description
                   "An arbitrary name for this endpoint.";
               }
               choice transport {
                 mandatory true;
                 description
                   "Selects between available transports. This is a
                    'choice' statement so as to support additional
                    transport options to be augmented in.";
                 case tls {
                   if-feature tls-initiate;
                   container tls {
                     description
                       "Specifies TLS-specific transport
                        configuration.";
                     leaf address {
                       type inet:host;
                       mandatory true;
                       description
                        "The IP address or hostname of the endpoint.
                         If a domain name is configured, then the
                         DNS resolution should happen on each usage
                         attempt.  If the the DNS resolution results
                         in multiple IP addresses, the IP addresses
                         will be tried according to local preference
                         order until a connection has been established
                         or until all IP addresses have failed.";
                     }
                     leaf port {
                       type inet:port-number;
                       default 443;
                       description
```

```
                        "The IP port for this endpoint. The RESTCONF
                         client will use the IANA-assigned well-known
                         port for 'https' (443) if no value is
                         specified.";
                    }
                    uses ts:tls-client-grouping {
                      refine "client-identity/auth-type" {
                        mandatory true;
                        description
                          "RESTCONF clients MUST pass some
                           authentication credentials.";
                      }
                    }
                  }
                } // end tls
              } // end transport
              container connection-type {
                description
                 "Indicates the kind of connection to use.";
                choice connection-type {
                  mandatory true;
                  description
                    "Selects between available connection types.";
                  case persistent-connection {
                    container persistent {
                      presence
                       "Indicates that a persistent connection is
                        to be maintained.";
                      description
                       "Maintain a persistent connection to the
                        RESTCONF server. If the connection goes down,
                        immediately start trying to reconnect to it,
                        using the reconnection strategy.  This
                        connection type minimizes any RESTCONF server
                        to RESTCONF client data-transfer delay, albeit
                        at the expense of holding resources longer.";
                      container keep-alives {
                        description
                          "Configures the keep-alive policy, to
                           proactively test the aliveness of the TLS
                           server.  An unresponsive TLS server will
                           be dropped after approximately max-attempts
                           * max-wait seconds.";
                        leaf max-wait {
                          type uint16 {
                            range "1..max";
                          }
                          units seconds;
```

```
                  default 30;
                  description
                   "Sets the amount of time in seconds after
                    which if no data has been received from
                    the TLS server, a TLS-level message will
                    be sent to test the aliveness of the TLS
                    server.";
                }
                leaf max-attempts {
                  type uint8;
                  default 3;
                  description
                   "Sets the maximum number of sequential
                    keep-alive messages that can fail to
                    obtain a response from the TLS server
                    before assuming the TLS server is no
                    longer alive.";
                }
              }
            }
          }
          case periodic-connection {
            container periodic {
              presence
               "Indicates that a periodic connection is to be
                maintained.";
              description
               "Periodically connect to the NETCONF server.
                The RESTCONF server should close the underlying
                TLS connection upon completing planned
                activities.

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

```
                                 + '(Z|[\+\-]\d{2}:\d{2})';
                      }
                      description
                        "Designates a timestamp before or after which
                         a series of periodic connections are
                         determined.  The periodic connections occur
                         at a whole multiple interval from the anchor
                         time.  For example, for an anchor time is 15
                         minutes past midnight and a period interval
                         of 24 hours, then a periodic connection will
                         occur 15 minutes past midnight everyday.";
                    }
                    leaf idle-timeout {
                      type uint16;
                      units "seconds";
                      default 120; // two minutes
                      description
                        "Specifies the maximum number of seconds
                         that the underlying TLS session may remain
                         idle. A TLS 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.";
                    }
                  }
                } // end periodic-connection
              } // end connection-type
            } // end 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).";
           }
         } // end reconnect-strategy
       } // end endpoint
     } // end endpoints
   } // end restconf-server
 } // end initiate

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

   leaf idle-timeout {
     type uint16;
     units "seconds";
     default 3600; // one hour
```

```
          description
            "Specifies the maximum number of seconds that an
             underlying TLS session may remain idle. A TLS 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 tls {
              if-feature tls-listen;
              container tls {
                description
                  "TLS-specific listening configuration for inbound
                   connections.";
                leaf address {
                  type inet:ip-address;
                  description
                   "The IP address to listen on for incoming call-
                    home connections.  The RESTCONF client will
                    listen on all configured interfaces if no
                    value is specified.  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 port {
                  type inet:port-number;
                  default 4336;
                  description
                   "The port number to listen on for call-home
```

```
                     connections. The RESTCONF client will listen
                     on the IANA-assigned well-known port for
                     'restconf-ch-tls' (4336) if no value is
                      specified.";
                 }
                 uses ts:tls-client-grouping {
                   refine "client-identity/auth-type" {
                     mandatory true;
                     description
                       "RESTCONF clients MUST pass some authentication
                        credentials.";
                   }
                 }
               }
             }
           } // end transport
         } // end endpoint
       } // end listen
     } // end restconf-client
   }
   <CODE ENDS>
```

3. The RESTCONF Server Model

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

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

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

3.1. Tree Diagram

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

[Note: '\' line wrapping for formatting only]

```
module: ietf-restconf-server
  +--rw restconf-server
     +--rw listen! {listen}?
     |  +--rw endpoint* [name]
```

```
        |      +--rw name         string
        |      +--rw (transport)
        |         +--:(tls) {tls-listen}?
        |            +--rw tls
        |               +--rw address?           inet:ip-address
        |               +--rw port?              inet:port-number
        |               +--rw server-identity
        |               |  +--rw (local-or-keystore)
        |               |     +--:(local) {local-keys-supported}?
        |               |     |  +--rw algorithm?
        |               |     |  |       asymmetric-key-encryption-algor\
   ithm-ref
        |               |     |  +--rw public-key?               binary
        |               |     |  +--rw private-key?              union
        |               |     |  +---x generate-hidden-key
        |               |     |  |  +---w input
        |               |     |  |     +---w algorithm
        |               |     |  |             asymmetric-key-encryption\
   -algorithm-ref
        |               |     |  +---x install-hidden-key
        |               |     |  |  +---w input
        |               |     |  |     +---w algorithm
        |               |     |  |     |       asymmetric-key-encryption\
   -algorithm-ref
        |               |     |  |     +---w public-key?    binary
        |               |     |  |     +---w private-key?   binary
        |               |     |  +--rw cert?
        |               |     |  |       end-entity-cert-cms
        |               |     |  +---n certificate-expiration
        |               |     |     +-- expiration-date
        |               |     |             yang:date-and-time
        |               |     +--:(keystore) {keystore-supported}?
        |               |        +--rw reference?
        |               |                ks:asymmetric-key-certificate-r\
   ef
        |               +--rw client-auth
        |               |  +--rw pinned-ca-certs?
        |               |  |       ta:pinned-certificates-ref
        |               |  |       {ta:x509-certificates}?
        |               |  +--rw pinned-client-certs?
        |               |  |       ta:pinned-certificates-ref
        |               |  |       {ta:x509-certificates}?
        |               |  +--rw cert-maps
        |               |     +--rw cert-to-name* [id]
        |               |        +--rw id            uint32
        |               |        +--rw fingerprint
        |               |        |       x509c2n:tls-fingerprint
        |               |        +--rw map-type       identityref
```

```
        |              |        +--rw name           string
        |              +--rw hello-params
        |                      {tls-server-hello-params-config}?
        |                 +--rw tls-versions
        |                 |  +--rw tls-version*   identityref
        |                 +--rw cipher-suites
        |                    +--rw cipher-suite*   identityref
        +--rw call-home! {call-home}?
           +--rw restconf-client* [name]
              +--rw name                  string
              +--rw endpoints
              |  +--rw endpoint* [name]
              |     +--rw name         string
              |     +--rw (transport)
              |        +--:(tls) {tls-call-home}?
              |           +--rw tls
              |              +--rw address            inet:host
              |              +--rw port?              inet:port-number
              |              +--rw server-identity
              |              |  +--rw (local-or-keystore)
              |              |     +--:(local) {local-keys-supported}?
              |              |     |  +--rw algorithm?
              |              |     |  |       asymmetric-key-encryption\
   -algorithm-ref
              |              |     |  +--rw public-key?
              |              |     |  |       binary
              |              |     |  +--rw private-key?
              |              |     |  |       union
              |              |     |  +---x generate-hidden-key
              |              |     |  |  +---w input
              |              |     |  |     +---w algorithm
              |              |     |  |             asymmetric-key-encr\
   yption-algorithm-ref
              |              |     |  +---x install-hidden-key
              |              |     |  |  +---w input
              |              |     |  |     +---w algorithm
              |              |     |  |     |       asymmetric-key-encr\
   yption-algorithm-ref
              |              |     |  |     +---w public-key?    binary
              |              |     |  |     +---w private-key?   binary
              |              |     |  +--rw cert?
              |              |     |  |       end-entity-cert-cms
              |              |     |  +---n certificate-expiration
              |              |     |     +-- expiration-date
              |              |     |             yang:date-and-time
              |              |     +--:(keystore) {keystore-supported}?
              |              |        +--rw reference?
              |              |                ks:asymmetric-key-certifi\
```

```
   cate-ref
             |              +--rw client-auth
             |              |  +--rw pinned-ca-certs?
             |              |  |       ta:pinned-certificates-ref
             |              |  |       {ta:x509-certificates}?
             |              |  +--rw pinned-client-certs?
             |              |  |       ta:pinned-certificates-ref
             |              |  |       {ta:x509-certificates}?
             |              |  +--rw cert-maps
             |              |     +--rw cert-to-name* [id]
             |              |        +--rw id             uint32
             |              |        +--rw fingerprint
             |              |        |       x509c2n:tls-fingerprint
             |              |        +--rw map-type       identityref
             |              |        +--rw name           string
             |              +--rw hello-params
             |                      {tls-server-hello-params-config}?
             |                 +--rw tls-versions
             |                 |  +--rw tls-version*   identityref
             |                 +--rw cipher-suites
             |                    +--rw cipher-suite*   identityref
             +--rw connection-type
             |  +--rw (connection-type)
             |     +--:(persistent-connection)
             |     |  +--rw persistent!
             |     |     +--rw keep-alives
             |     |        +--rw max-wait?       uint16
             |     |        +--rw max-attempts?   uint8
             |     +--:(periodic-connection)
             |        +--rw periodic!
             |           +--rw period?         uint16
             |           +--rw anchor-time?    yang:date-and-time
             |           +--rw idle-timeout?   uint16
             +--rw reconnect-strategy
                +--rw start-with?     enumeration
                +--rw max-attempts?   uint8
```

3.2. Example Usage

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

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

[Note: '\' line wrapping for formatting only]

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

     <!-- call home to a RESTCONF client with two endpoints -->
     <call-home>
       <restconf-client>
         <name>config-manager</name>
         <endpoints>
           <endpoint>
             <name>east-data-center</name>
             <tls>
```

```
               <address>22.33.44.55</address>
               <server-identity>
                 <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
   crypto-types">ct:rsa2048</algorithm>
                 <private-key>base64encodedvalue==</private-key>
                 <public-key>base64encodedvalue==</public-key>
                 <cert>base64encodedvalue==</cert>
               </server-identity>
               <client-auth>
                 <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
   inned-ca-certs>
                 <pinned-client-certs>explicitly-trusted-client-certs</\
   pinned-client-certs>
                 <cert-maps>
                   <cert-to-name>
                     <id>1</id>
                     <fingerprint>11:0A:05:11:00</fingerprint>
                     <map-type>x509c2n:san-any</map-type>
                   </cert-to-name>
                   <cert-to-name>
                     <id>2</id>
                     <fingerprint>B3:4F:A1:8C:54</fingerprint>
                     <map-type>x509c2n:specified</map-type>
                     <name>scooby-doo</name>
                   </cert-to-name>
                 </cert-maps>
               </client-auth>
             </tls>
           </endpoint>
           <endpoint>
             <name>west-data-center</name>
             <tls>
               <address>33.44.55.66</address>
               <server-identity>
                 <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-\
   crypto-types">ct:rsa2048</algorithm>
                 <private-key>base64encodedvalue==</private-key>
                 <public-key>base64encodedvalue==</public-key>
                 <cert>base64encodedvalue==</cert>
               </server-identity>
               <client-auth>
                 <pinned-ca-certs>explicitly-trusted-client-ca-certs</p\
   inned-ca-certs>
                 <pinned-client-certs>explicitly-trusted-client-certs</\
   pinned-client-certs>
                 <cert-maps>
                   <cert-to-name>
                     <id>1</id>
```

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


3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-restconf-server@2018-10-22.yang"
module ietf-restconf-server {
  yang-version 1.1;

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

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

  import ietf-inet-types {
```

```
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types";
     }

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

     import ietf-tls-server {
       prefix ts;
       revision-date 2018-10-22; // stable grouping definitions
       reference
         "RFC ZZZZ: YANG Groupings for TLS Clients and TLS Servers";
     }

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

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

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.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.

       Copyright (c) 2017 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: RESTCONF Client and Server Models";
     }


     // Features

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

     feature tls-listen {
       if-feature listen;
       description
        "The 'tls-listen' feature indicates that the RESTCONF server
         supports opening a port to listen for incoming RESTCONF
         client connections.  This feature exists as TLS might not
         be a mandatory to implement transport in the future.";
       reference
        "RFC 8040: RESTCONF Protocol";
     }

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

     feature tls-call-home {
       if-feature call-home;
       description
        "The 'tls-call-home' feature indicates that the RESTCONF
         server supports initiating connections to RESTCONF clients.
         This feature exists as TLS might not be a mandatory to
         implement transport in the future.";
```

```
       reference
        "RFC 8071: NETCONF Call Home and RESTCONF Call Home";
     }

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

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

       container listen {
         if-feature listen;
         presence "Enables server to listen for TCP connections";
         description "Configures listen behavior";
         list endpoint {
           key name;
           min-elements 1;
           description
             "List of endpoints to listen for RESTCONF connections.";
           leaf name {
             type string;
             description
               "An arbitrary name for the RESTCONF listen endpoint.";
           }
           choice transport {
             mandatory true;
             description
               "Selects between available transports. This is a
                'choice' statement so as to support additional
                transport options to be augmented in.";
             case tls {
               if-feature tls-listen;
               container tls {
                 description
                   "TLS-specific listening configuration for inbound
                    connections.";
                 leaf address {
                   type inet:ip-address;
                 description
                   "The IP address to listen on for incoming
                    connections.  The RESTCONF server will listen
                    on all configured interfaces if no value is
                    specified.  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 port {
                  type inet:port-number;
                  default 443;
                  description
                   "The local port number to listen on.  If no value
                    is specified, the IANA-assigned port value for
                    'https' (443) is used.";
                }
                uses ts:tls-server-grouping {
                  refine "client-auth" {
                    must 'pinned-ca-certs or pinned-client-certs';
                    description
                      "RESTCONF servers MUST be able to validate
                       clients.";
                  }
                  augment "client-auth" {
                    description
                      "Augments in the cert-to-name structure,
                       so the RESTCONF server can map TLS-layer
                       client certificates to RESTCONF usernames.";
                    container cert-maps {
                      uses x509c2n:cert-to-name;
                      description
                       "The cert-maps container is used by a TLS-
                        based RESTCONF server to map the RESTCONF
                        client's presented X.509 certificate to
                        a RESTCONF username.  If no matching and
                        valid cert-to-name list entry can be found,
                        then the RESTCONF server MUST close the
                        connection, and MUST NOT accept RESTCONF
                        messages over it.";
                      reference
                        "RFC 7407: A YANG Data Model for SNMP
                                   Configuration.";
                    }
                  }
                }
              } // end tls container
            } // end tls case
          } // end transport
        } // end endpoint
      } // end listen

      container call-home {
        if-feature call-home;
```

```
          presence "Enables server to initiate TCP connections";
          description "Configures call-home behavior";
          list restconf-client {
            key name;
            min-elements 1;
            description
              "List of RESTCONF clients the RESTCONF server is to
               initiate call-home connections to in parallel.";
            leaf name {
              type string;
              description
                "An arbitrary name for the remote RESTCONF client.";
            }
            container endpoints {
              description
                "Container for the list of endpoints.";
              list endpoint {
                key name;
                min-elements 1;
                ordered-by user;
                description
                  "User-ordered list of endpoints for this RESTCONF
                   client.  Defining more than one enables high-
                   availability.";
                leaf name {
                  type string;
                  description
                    "An arbitrary name for this endpoint.";
                }
                choice transport {
                  mandatory true;
                  description
                    "Selects between available transports. This is a
                     'choice' statement so as to support additional
                     transport options to be augmented in.";
                  case tls {
                    if-feature tls-call-home;
                    container tls {
                      description
                        "Specifies TLS-specific call-home transport
                         configuration.";
                      leaf address {
                        type inet:host;
                        mandatory true;
                        description
                         "The IP address or hostname of the endpoint.
                          If a domain name is configured, then the
                          DNS resolution should happen on each usage
```

```
                      attempt.  If the DNS resolution results in
                      multiple IP addresses, the IP addresses will
                      be tried according to local preference order
                      until a connection has been established or
                      until all IP addresses have failed.";
                  }
                  leaf port {
                    type inet:port-number;
                    default 4336;
                    description
                     "The IP port for this endpoint.  The RESTCONF
                      server will use the IANA-assigned well-known
                      port for 'restconf-ch-tls' (4336) if no value
                      is specified.";
                  }
                  uses ts:tls-server-grouping {
                    refine "client-auth" {
                     must 'pinned-ca-certs or pinned-client-certs';
                     description
                      "RESTCONF servers MUST be able to validate
                       clients.";
                    }
                    augment "client-auth" {
                      description
                       "Augments in the cert-to-name structure,
                        so the RESTCONF server can map TLS-layer
                        client certificates to RESTCONF usernames.";
                      container cert-maps {
                        uses x509c2n:cert-to-name;
                        description
                         "The cert-maps container is used by a
                          TLS-based RESTCONF server to map the
                          RESTCONF client's presented X.509
                          certificate to a RESTCONF username. If
                          no matching and valid cert-to-name list
                          entry can be found, then the RESTCONF
                          server MUST close the connection, and
                          MUST NOT accept RESTCONF messages over
                          it.";
                        reference
                          "RFC 7407: A YANG Data Model for SNMP
                           Configuration.";
                      }
                    }
                  }
                }
              }
            } // end transport
```

```
            } // end endpoint
          } // end endpoints
          container connection-type {
            description
             "Indicates the RESTCONF client's preference for how the
              RESTCONF server's connection is maintained.";
            choice connection-type {
              mandatory true;
              description
                "Selects between available connection types.";
              case persistent-connection {
                container persistent {
                  presence
                   "Indicates that a persistent connection is to be
                    maintained.";
                  description
                   "Maintain a persistent connection to the RESTCONF
                    client. If the connection goes down, immediately
                    start trying to reconnect to it, using the
                    reconnection strategy.

                    This connection type minimizes any RESTCONF
                    client to RESTCONF server data-transfer delay,
                    albeit at the expense of holding resources
                    longer.";
                  container keep-alives {
                    description
                      "Configures the keep-alive policy, to
                       proactively test the aliveness of the TLS
                       client.  An unresponsive TLS client will
                       be dropped after approximately (max-attempts
                       * max-wait) seconds.";
                    reference
                      "RFC 8071: NETCONF Call Home and RESTCONF
                                 Call Home, Section 4.1, item S7";
                    leaf max-wait {
                      type uint16 {
                        range "1..max";
                      }
                      units seconds;
                      default 30;
                      description
                       "Sets the amount of time in seconds after
                        which if no data has been received from
                        the TLS client, a TLS-level message will
                        be sent to test the aliveness of the TLS
                        client.";
                    }
```

```
                    leaf max-attempts {
                      type uint8;
                      default 3;
                      description
                       "Sets the maximum number of sequential keep-
                        alive messages that can fail to obtain a
                        response from the TLS client before assuming
                        the TLS client is no longer alive.";
                    }
                  }
                }
              }
              case periodic-connection {
                container periodic {
                  presence
                   "Indicates that a periodic connection is to be
                    maintained.";
                  description
                   "Periodically connect to the RESTCONF client.  The
                    RESTCONF client should close the underlying TLS
                    connection upon completing planned activities.

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

```
                  leaf idle-timeout {
                    type uint16;
                    units "seconds";
                    default 120; // two minutes
                    description
                      "Specifies the maximum number of seconds that
                       the underlying TLS session may remain idle.
                       A TLS session will be dropped if it is idle
                       for an interval longer than this number of
                       seconds.  If set to zero, then the server
                       will never drop a session because it is idle.";
                  }
                }
              }
            }
          }
          container reconnect-strategy {
            description
             "The reconnection strategy directs how a RESTCONF server
              reconnects to a RESTCONF client after discovering its
              connection to the client has dropped, even if due to a
              reboot.  The RESTCONF server starts with the specified
              endpoint and tries to connect to it max-attempts times
              before trying the next endpoint in the list (round
              robin).";
            leaf start-with {
              type enumeration {
                enum first-listed {
                  description
                    "Indicates that reconnections should start with
                     the first endpoint listed.";
                }
                enum last-connected {
                  description
                    "Indicates that reconnections should start with
                     the endpoint last connected to.  If no previous
                     connection has ever been established, then the
                     first endpoint configured is used.   RESTCONF
                     servers SHOULD be able to remember the last
                     endpoint connected to across reboots.";
                }
                enum random-selection {
                  description
                    "Indicates that reconnections should start with
                     a random endpoint.";
                }
              }
              default first-listed;
```

```
              description
               "Specifies which of the RESTCONF client's endpoints
                the RESTCONF server should start with when trying
                to connect to the RESTCONF client.";
            }
            leaf max-attempts {
              type uint8 {
                range "1..max";
              }
              default 3;
              description
               "Specifies the number times the RESTCONF server tries
                to connect to a specific endpoint before moving on to
                the next endpoint in the list (round robin).";
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

4. Security Considerations

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

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

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

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

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

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

NONE

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

NONE

5. IANA Considerations

5.1. The IETF XML Registry

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

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

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

5.2. The YANG Module Names Registry

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

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

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

6. References

6.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-06 (work in progress), September 2018.

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", draft-ietf-netconf-tls-client-server-07 (work in progress), September 2018.

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

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7407] Bjorklund, M. and J. Schoenwaelder, "A YANG Data Model for SNMP Configuration", RFC 7407, DOI 10.17487/RFC7407, December 2014, <https://www.rfc-editor.org/info/rfc7407>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

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

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

6.2. Informative References

[I-D.ietf-netconf-netconf-client-server] Watsen, K., "NETCONF Client and Server Models", draft-ietf-netconf-netconf-client-server-07 (work in progress), September 2018.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

Appendix A. Change Log

A.1. 00 to 01

- Renamed "keychain" to "keystore".

A.2. 01 to 02

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

A.3. 02 to 03

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

A.4. 03 to 04

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

A.5. 04 to 05

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

A.6. 05 to 06

- Fixed change log missing section issue.

- Updated examples to match latest updates to the crypto-types, trust-anchors, and keystore drafts.

- Reduced line length of the YANG modules to fit within 69 columns.

A.7. 06 to 07

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

- Added "random-selection" for reconnection-strategy's "starts-with" enum.

- Replaced "connection-type" choice default (persistent) with "mandatory true".

- Reduced the periodic-connection's "idle-timeout" from 5 to 2 minutes.

- Replaced reconnect-timeout with period/anchor-time combo.

A.8. 07 to 08

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

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.

Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2019

E. Voit
R. Rahman
E. Nilsen-Nygaard
Cisco Systems
A. Clemm
Huawei
A. Bierman
YumaWorks

October 19, 2018

Dynamic subscription to YANG Events and Datastores over RESTCONF
draft-ietf-netconf-restconf-notif-09

Abstract

This document provides a RESTCONF binding to the dynamic subscription capability of both subscribed notifications and YANG-Push.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. Dynamic Subscriptions 3
3.1. Transport Connectivity 4
3.2. Discovery 4
3.3. RESTCONF RPCs and HTTP Status Codes 4
3.4. Call Flow for Server-Sent Events (SSE) 6
4. QoS Treatment 8
5. Notification Messages 8
6. YANG Tree 8
7. YANG module 8
8. IANA Considerations 10
9. Security Considerations 11
10. Acknowledgments 11
11. References 11
11.1. Normative References 11
11.2. Informative References 13
Appendix A. Examples 14
A.1. Dynamic Subscriptions 14
A.1.1. Establishing Dynamic Subscriptions 14
A.1.2. Modifying Dynamic Subscriptions 17
A.1.3. Deleting Dynamic Subscriptions 18
A.2. Subscription State Notifications 19
A.2.1. subscription-modified 19
A.2.2. subscription-completed, subscription-resumed, and replay-complete 20
A.2.3. subscription-terminated and subscription-suspended . 20
A.3. Filter Example 21
Appendix B. Changes between revisions 22
Authors' Addresses 24

1. Introduction

Mechanisms to support event subscription and push are defined in [I-D.draft-ietf-netconf-subscribed-notifications]. Enhancements to [I-D.draft-ietf-netconf-subscribed-notifications] which enable YANG datastore subscription and push are defined in [I-D.ietf-netconf-yang-push]. This document provides a transport specification for dynamic subscriptions over RESTCONF [RFC8040]. Driving these requirements is [RFC7923].

The streaming of notifications encapsulating the resulting information push is done via the mechanism described in section 6.3 of [RFC8040].

2. Terminology

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].

The following terms use the definitions from [I-D.draft-ietf-netconf-subscribed-notifications]: dynamic subscription, event stream, notification message, publisher, receiver, subscriber, and subscription.

Other terms reused include datastore, which is defined in [RFC8342], and HTTP2 stream which maps to the definition of "stream" within [RFC7540], Section 2.

[note to the RFC Editor - please replace XXXX within this document with the number of this document]

3. Dynamic Subscriptions

This section provides specifics on how to establish and maintain dynamic subscriptions over RESTCONF [RFC8040]. Subscribing to event streams is accomplished in this way via RPCs defined within [I-D.draft-ietf-netconf-subscribed-notifications] Section 2.4, the RPCs are done via RESTCONF POSTs. YANG datastore subscription is accomplished via augmentations to [I-D.draft-ietf-netconf-subscribed-notifications] as described within [I-D.ietf-netconf-yang-push] Section 4.4.

As described in [RFC8040] Section 6.3, a GET needs to be made against a specific URI on the publisher. Subscribers cannot pre-determine the URI against which a subscription might exist on a publisher, as the URI will only exist after the "establish-subscription" RPC has been accepted. Therefore, the POST for the "establish-subscription" RPC replaces the GET request for the "location" leaf which is used in [RFC8040] to obtain the URI. The subscription URI will be determined and sent as part of the response to the "establish-subscription" RPC, and a subsequent GET to this URI will be done in order to start the flow of notification messages back to the subscriber. A subscription does not move to the active state as per Section 2.4.1. of [I-D.draft-ietf-netconf-subscribed-notifications] until the GET is received.

3.1. Transport Connectivity

For a dynamic subscription, where a RESTCONF session doesn't already exist, a new RESTCONF session is initiated from the subscriber.

As stated in Section 2.1 of [RFC8040], a subscriber MUST establish the HTTP session over TLS [RFC5246] in order to secure the content in transit.

Without the involvement of additional protocols, HTTP sessions by themselves do not allow for a quick recognition of when the communication path has been lost with the publisher. Where quick recognition of the loss of a publisher is required, a subscriber SHOULD use a TLS heartbeat [RFC6520], just from receiver to publisher, to track HTTP session continuity.

Loss of the heartbeat MUST result in any subscription related TCP sessions between those endpoints being torn down. A subscriber can then attempt to re-establish the dynamic subscription by using the procedure described in Section 3.

3.2. Discovery

Subscribers can learn what event streams a RESTCONF server supports by querying the "streams" container of ietf-subscribed-notification.yang in [I-D.draft-ietf-netconf-subscribed-notifications]. Support for the "streams" container of ietf-restconf-monitoring.yang in [RFC8040] is not required.

Subscribers can learn what datastores a RESTCONF server supports by following [I-D.draft-ietf-netconf-nmda-restconf].

3.3. RESTCONF RPCs and HTTP Status Codes

Specific HTTP responses codes as defined in [RFC7231] section 6 will indicate the result of RESTCONF RPC requests with publisher. An HTTP status code of 200 is the proper response to any successful RPC defined within [I-D.draft-ietf-netconf-subscribed-notifications] or [I-D.ietf-netconf-yang-push].

If a publisher fails to serve the RPC request for one of the reasons indicated in [I-D.draft-ietf-netconf-subscribed-notifications] Section 2.4.6 or [I-D.ietf-netconf-yang-push] Appendix A, this will be indicated by "406" status code transported in the HTTP response.

When a "406" status code is returned, the RPC reply MUST include an "rpc-error" element per [RFC8040] Section 7.1 with the following parameter values:

o an "error-type" node of "application".

o an "error-tag" node of "operation-failed".

o an "error-app-tag" node with the value being a string that corresponds to an identity associated with the error, as defined in [I-D.draft-ietf-netconf-subscribed-notifications] section 2.4.6 for general subscriptions, and [I-D.ietf-netconf-yang-push] Appendix A.1, for datastore subscriptions. The tag to use depends on the RPC for which the error occurred. Viable errors for different RPCs are as follows:

RPC	select an identity with a base
establish-subscription	establish-subscription-error
modify-subscription	modify-subscription-error
delete-subscription	delete-subscription-error
kill-subscription	kill-subscription-error
resynch-subscription	resynch-subscription-error

Each error identity will be inserted as the "error-app-tag" using JSON encoding following the form <modulename>:<identityname>. An example of such as valid encoding would be "ietf-subscribed-notifications:no-such-subscription".

In case of error responses to an "establish-subscription" or "modify-subscription" request there is the option of including an "error-info" node. This node may contain hints for parameter settings that might lead to successful RPC requests in the future. Following are the yang-data structures which may be returned:

```
establish-subscription returns hints in yang-data structure
---------------------- ------------------------------------
target: event stream   establish-subscription-stream-error-info
target: datastore      establish-subscription-datastore-error-info

modify-subscription    returns hints in yang-data structure
---------------------- ------------------------------------
target: event stream   modify-subscription-stream-error-info
target: datastore      modify-subscription-datastore-error-info
```

The yang-data included within "error-info" SHOULD NOT include the optional leaf "error-reason", as such a leaf would be redundant with information that is already placed within the "error-app-tag".

In case of an rpc error as a result of a "delete-subscription", a "kill-subscription", or a "resynch-subscription" request, no "error-info" needs to be included, as the "subscription-id" is the only RPC input parameter and no hints regarding this RPC input parameters need to be provided.

Note that "error-path" [RFC8040] does not need to be included with the "rpc-error" element, as subscription errors are generally associated with the choice of RPC input parameters.

3.4. Call Flow for Server-Sent Events (SSE)

The call flow is defined in Figure 1. The logical connections denoted by (a) and (b) can be a TCP connection or an HTTP2 stream (multiple HTTP2 streams can be carried in one TCP connection). Requests to [I-D.draft-ietf-netconf-subscribed-notifications] or [I-D.ietf-netconf-yang-push] augmented RPCs are sent on a connection indicated by (a). A successful "establish-subscription" will result in an RPC response returned with both a subscription identifier which uniquely identifies a subscription, as well as a URI which uniquely identifies the location of subscription on the publisher (b). This URI is defined via the "uri" leaf the Data Model in Section 7.

An HTTP GET is then sent on a separate logical connection (b) to the URI on the publisher. This initiates the publisher to initiate the flow of notification messages which are sent in SSE [W3C-20150203] as a response to the GET.

```
   +--------------+                             +--------------+
   |  Subscriber  |                             |   Publisher  |
   |              |                             |              |
   |    Logical   |                             |     Logical  |
   |  Connection  |                             |   Connection |
   |  (a)  (b)    |                             |    (a)  (b)  |
   +--------------+                             +--------------+
       | RESTCONF POST (RPC:establish-subscription)   |
       |--------------------------------------------->|
       |                          HTTP 200 OK (ID,URI)|
       |<---------------------------------------------|
       |    |HTTP GET (URI)                                |
       |    |--------------------------------------------->|
       |    |                                   HTTP 200 OK|
       |    |<---------------------------------------------|
       |    |                           SSE (notif-message)|
       |    |<---------------------------------------------|
       | RESTCONF POST (RPC:modify-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                              HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |                   SSE (subscription-modified)|
       |    |<------------------------------------------(c)|
       |    |                           SSE (notif-message)|
       |    |<---------------------------------------------|
       | RESTCONF POST (RPC:delete-subscription)      |    |
       |--------------------------------------------->|    |
       |    |                              HTTP 200 OK|    |
       |<---------------------------------------------|    |
       |    |                                              |
       |    |
```

Figure 1: Dynamic with server-sent events

Additional requirements for dynamic subscriptions over SSE include:

- o All subscription state notifications from a publisher MUST be returned in a separate SSE message used by the subscription to which the state change refers.
- o Subscription RPCs MUST NOT use the connection currently providing notification messages for that subscription.
- o In addition to an RPC response for a "modify-subscription" RPC traveling over (a), a "subscription-modified" state change notification must be sent within (b). This allows the receiver to know exactly when the new terms of the subscription have been applied to the notification messages. See arrow (c).

A publisher MUST terminate a subscription in the following cases:

- o Receipt of a "delete-subscription" or a "kill-subscription" RPC for that subscription.

- o Loss of TLS heartbeat

A publisher MAY terminate a subscription at any time as stated in [I-D.draft-ietf-netconf-subscribed-notifications] Section 1.3

4. QoS Treatment

To meet subscription quality of service promises, the publisher MUST take any existing subscription "dscp" and apply it to the DSCP marking in the IP header.

In addition, where HTTP2 transport is available to a notification message queued for transport to a receiver, the publisher MUST:

- o take any existing subscription "priority", as specified by the "dscp" leaf node in [I-D.draft-ietf-netconf-subscribed-notifications], and copy it into the HTTP2 stream priority, [RFC7540] section 5.3, and

- o take any existing subscription "dependency", as specified by the "dependency" leaf node in [I-D.draft-ietf-netconf-subscribed-notifications], and use the HTTP2 stream for the parent subscription as the HTTP2 stream dependency, [RFC7540] section 5.3.1, of the dependent subscription.

5. Notification Messages

Notification messages transported over RESTCONF will be encoded according to [RFC8040], section 6.4.

6. YANG Tree

The YANG model defined in Section 7 has one leaf augmented into four places of [I-D.draft-ietf-netconf-subscribed-notifications], plus two identities. As the resulting full tree is large, it will only be inserted at later stages of this document.

7. YANG module

This module references [I-D.draft-ietf-netconf-subscribed-notifications].

```
<CODE BEGINS> file "ietf-restconf-subscribed-notifications@2018-10-19.yang"
module ietf-restconf-subscribed-notifications {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-restconf-subscribed-notifications";

  prefix rsn;

  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:   Eric Voit
               <mailto:evoit@cisco.com>

     Editor:   Alexander Clemm
               <mailto:ludwig@clemm.org>

     Editor:   Reshad Rahman
               <mailto:rrahman@cisco.com>";

  description
    "Defines RESTCONF as a supported transport for subscribed
    event notifications.

    Copyright (c) 2018 IETF Trust and the persons identified as authors
    of the code.  All rights reserved.

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

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

  revision 2018-10-19 {
    description
      "Initial version";
```

```
    reference
      "RFC XXXX: RESTCONF Transport for Event Notifications";
  }

  grouping uri {
    description
      "Provides a reusable description of a URI.";
    leaf uri {
      type inet:uri;
      config false;
      description
        "Location of a subscription specific URI on the publisher.";
    }
  }

  augment "/sn:establish-subscription/sn:output" {
    description
      "This augmentation allows RESTCONF specific parameters for a
       response to a publisher's subscription request.";
    uses uri;
  }

  augment "/sn:subscriptions/sn:subscription" {
    description
      "This augmentation allows RESTCONF specific parameters to be
       exposed for a subscription.";
    uses uri;
  }

  augment "/sn:subscription-modified" {
    description
      "This augmentation allows RESTCONF specific parameters to be included
       part of the notification that a subscription has been modified.";
    uses uri;
  }
}
<CODE ENDS>
```

8. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-restconf-subscribed-notifications
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 [RFC6020]:

Name: ietf-restconf-subscribed-notifications
Namespace: urn:ietf:params:xml:ns:yang:ietf-restconf-subscribed-notifications
Prefix: rsn
Reference: RFC XXXX: RESTCONF Transport for Event Notifications

9. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management transports 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 one new data node 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, get-config, or notification) to this data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

Container: "/subscriptions"

o "uri": leaf will show where subscribed resources might be located on a publisher. Access control must be set so that only someone with proper access permissions, and perhaps even HTTP session has the ability to access this resource.

10. Acknowledgments

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from: Ambika Prasad Tripathy, Alberto Gonzalez Prieto, Susan Hares, Tim Jenkins, Balazs Lengyel, Kent Watsen, Michael Scharf, Guangying Zheng, Martin Bjorklund and Qin Wu.

11. References

11.1. Normative References

[I-D.draft-ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Gonzalez Prieto, A., Tripathy, A., and E. Nilsen-Nygaard, "Custom Subscription to Event Streams", draft-ietf-netconf-subscribed-notifications-13 (work in progress), April 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Gonzalez Prieto, A., Prasad Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", March 2017, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

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

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

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

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <https://www.rfc-editor.org/info/rfc6520>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

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

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[W3C-20150203] "Server-Sent Events, World Wide Web Consortium CR CR-eventsource-20121211", February 2015, <https://www.w3.org/TR/2015/REC-eventsource-20150203/>.

11.2. Informative References

[I-D.draft-ietf-netconf-netconf-event-notifications] Clemm, Alexander., Voit, Eric., Gonzalez Prieto, Alberto., Nilsen-Nygaard, E., and A. Tripathy, "NETCONF support for event notifications", May 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-netconf-event-notifications/>.

[I-D.draft-ietf-netconf-nmda-restconf] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "RESTCONF Extensions to Support the Network Management Datastore Architecture", April 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-nmda-restconf/>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <https://www.rfc-editor.org/info/rfc7923>.

[RFC8347] Liu, X., Ed., Kyparlis, A., Parikh, R., Lindem, A., and M. Zhang, "A YANG Data Model for the Virtual Router Redundancy Protocol (VRRP)", RFC 8347, DOI 10.17487/RFC8347, March 2018, <https://www.rfc-editor.org/info/rfc8347>.

[XPATH] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

Appendix A. Examples

This section is non-normative. To allow easy comparison, this section mirrors the functional examples shown with NETCONF over XML within [I-D.draft-ietf-netconf-netconf-event-notifications]. In addition, HTTP2 vs HTTP1.1 headers are not shown as the contents of the JSON encoded objects are identical within.

A.1. Dynamic Subscriptions

A.1.1. Establishing Dynamic Subscriptions

The following figure shows two successful "establish-subscription" RPC requests as per [I-D.draft-ietf-netconf-subscribed-notifications]. The first request is given a subscription identifier of 22, the second, an identifier of 23.

```
  +------------+                 +-----------+
  | Subscriber |                 | Publisher |
  +------------+                 +-----------+
        |                               |
        |establish-subscription         |
        |------------------------------>|  (a)
        |     HTTP 200 OK, id#22, URI#1 |
        |<------------------------------|  (b)
        |GET (URI#1)                    |
        |------------------------------>|  (c)
        | HTTP 200 OK,notif-mesg (id#22)|
        |<------------------------------|
        |                               |
        |                               |
        |establish-subscription         |
        |------------------------------>|
        |      HTTP 200 OK, id#23, URI#2|
        |<------------------------------|
        |GET (URI#2)                    |
        |------------------------------>|
        |                               |
        |                               |
        |             notif-mesg (id#22)|
        |<------------------------------|
        | HTTP 200 OK,notif-mesg (id#23)|
        |<------------------------------|
        |                               |
```

Figure 2: Multiple subscriptions over RESTCONF/HTTP

To provide examples of the information being transported, example messages for interactions in Figure 2 are detailed below:

```
POST /restconf/operations/ietf-subscribed-notifications:establish-subscription

{
   "ietf-subscribed-notifications:input": {
      "stream": "NETCONF",
      "stream-xpath-filter": "/example-module:foo/",
      "dscp": "10"
   }
}
```

Figure 3: establish-subscription request (a)

As publisher was able to fully satisfy the request, the publisher sends the subscription identifier of the accepted subscription, and the URI:

```
HTTP status code - 200

{
   "id": "22",
   "uri": "https://example.com/restconf/subscriptions/22"
}
```

Figure 4: establish-subscription success (b)

Upon receipt of the successful response, the subscriber does a GET the provided URI to start the flow of notification messages. When the publisher receives this, the subscription is moved to the active state (c).

```
GET /restconf/subscriptions/22
```

Figure 5: establish-subscription subsequent POST

While not shown in Figure 2, if the publisher had not been able to fully satisfy the request, or subscriber has no authorization to establish the subscription, the publisher would have sent an RPC error response. For instance, if the "dscp" value of 10 asserted by the subscriber in Figure 3 proved unacceptable, the publisher may have returned:

```
HTTP status code - 406

{ "ietf-restconf:errors" : {
    "error" : [
      {
        "error-type": "application",
        "error-tag": "operation-failed",
        "error-severity": "error",
        "error-app-tag":
            "ietf-subscribed-notifications:dscp-unavailable"
      }
    ]
  }
}
```

Figure 6: an unsuccessful establish subscription

The subscriber can use this information in future attempts to establish a subscription.

A.1.2. Modifying Dynamic Subscriptions

An existing subscription may be modified. The following exchange shows a negotiation of such a modification via several exchanges between a subscriber and a publisher. This negotiation consists of a failed RPC modification request/response, followed by a successful one.

```
+------------+                  +-----------+
| Subscriber |                  | Publisher |
+------------+                  +-----------+
      |                               |
      |  notification message (id#23)|
      |<------------------------------|
      |                               |
      |modify-subscription (id#23)    |
      |------------------------------>|  (d)
      |    HTTP 406 error (with hint)|
      |<------------------------------|  (e)
      |                               |
      |modify-subscription (id#23)    |
      |------------------------------>|
      |                  HTTP 200 OK |
      |<------------------------------|
      |                               |
      |            notif-mesg (id#23)|
      |<------------------------------|
      |                               |
```

Figure 7: Interaction model for successful subscription modification

If the subscription being modified in Figure 7 is a datastore subscription as per [I-D.ietf-netconf-yang-push], the modification request made in (d) may look like that shown in Figure 8. As can be seen, the modifications being attempted are the application of a new xpath filter as well as the setting of a new periodic time interval.

```
POST /restconf/operations/ietf-subscribed-notifications:modify-subscription

{
 "ietf-subscribed-notifications:input": {
    "id": "23",
    "ietf-yang-push:datastore-xpath-filter": "/example-module:foo/example-module
:bar",
    "ietf-yang-push:periodic": {
       "ietf-yang-push:period": "500"
    }
  }
}
```

Figure 8: Subscription modification request (c)

If the publisher can satisfy both changes, the publisher sends a positive result for the RPC. If the publisher cannot satisfy either of the proposed changes, the publisher sends an RPC error response (e). The following is an example RPC error response for (e) which includes a hint. This hint is an alternative time period value which might have resulted in a successful modification:

```
HTTP status code - 406

{ "ietf-restconf:errors" : {
    "error" : [
      "error-type": "application",
      "error-tag": "operation-failed",
      "error-severity": "error",
      "error-app-tag": "ietf-yang-push:period-unsupported",
      "error-info": {
        "ietf-yang-push":
        "modify-subscription-datastore-error-info": {
           "period-hint": "3000"
        }
      }
    ]
  }
}
```

Figure 9: Modify subscription failure with Hint (e)

A.1.3. Deleting Dynamic Subscriptions

The following demonstrates deleting a subscription. This subscription may have been to either a stream or a datastore.

```
POST /restconf/operations/ietf-subscribed-notifications:delete-subscription

{
 "delete-subscription": {
    "id": "22"
 }
}
```

Figure 10: Delete subscription

If the publisher can satisfy the request, the publisher replies with success to the RPC request.

If the publisher cannot satisfy the request, the publisher sends an error-rpc element indicating the modification didn't work. Figure 11 shows a valid response for existing valid subscription identifier, but that subscription identifier was created on a different transport session:

```
HTTP status code - 406

{
  "ietf-restconf:errors" : {
    "error" : [
      "error-type": "application",
      "error-tag": "operation-failed",
      "error-severity": "error",
      "error-app-tag":
         "ietf-subscribed-notifications:no-such-subscription"
    ]
  }
}
```

Figure 11: Unsuccessful delete subscription

A.2. Subscription State Notifications

A publisher will send subscription state notifications according to the definitions within [I-D.draft-ietf-netconf-subscribed-notifications]).

A.2.1. subscription-modified

A "subscription-modified" encoded in JSON would look like:

```
{
  "ietf-restconf:notification" : {
    "eventTime": "2007-09-01T10:00:00Z",
    "ietf-subscribed-notifications:subscription-modified": {
      "id": "39",
      "uri": "https://example.com/restconf/subscriptions/22"
      "stream-xpath-filter": "/example-module:foo",
      "stream": {
         "ietf-netconf-subscribed-notifications" : "NETCONF"
      }
    }
  }
}
```


Figure 12: subscription-modified subscription state notification

A.2.2. subscription-completed, subscription-resumed, and replay-complete

A "subscription-completed" would look like:


```
{
  "ietf-restconf:notification" : {
    "eventTime": "2007-09-01T10:00:00Z",
    "ietf-subscribed-notifications:subscription-completed": {
      "id": "39",
    }
  }
}
```


Figure 13: subscription-completed notification in JSON

The "subscription-resumed" and "replay-complete" are virtually identical, with "subscription-completed" simply being replaced by "subscription-resumed" and "replay-complete".

A.2.3. subscription-terminated and subscription-suspended

A "subscription-terminated" would look like:

```
{
  "ietf-restconf:notification" : {
    "eventTime": "2007-09-01T10:00:00Z",
    "ietf-subscribed-notifications:subscription-terminated": {
      "id": "39",
      "error-id": "suspension-timeout"
    }
  }
}
```


Figure 14: subscription-terminated subscription state notification

The "subscription-suspended" is virtually identical, with "subscription-terminated" simply being replaced by "subscription-suspended".

A.3. Filter Example

This section provides an example which illustrate the method of filtering event record contents. The example is based on the YANG notification "vrrp-protocol-error-event" as defined per the ietf-vrrp.yang module within [RFC8347]. Event records based on this specification which are generated by the publisher might appear as:


```
data: {
data:   "ietf-restconf:notification" : {
data:     "eventTime" : "2018-09-14T08:22:33.44Z",
data:     "ietf-vrrp:vrrp-protocol-error-event" : {
data:       "protocol-error-reason" : "checksum-error"
data:     }
data:   }
data: }
```


Figure 15: RFC 8347 (VRRP) - Example Notification

Suppose a subscriber wanted to establish a subscription which only passes instances of event records where there is a "checksum-error" as part of a VRRP protocol event. Also assume the publisher places such event records into the NETCONF stream. To get a continuous series of matching event records, the subscriber might request the application of an XPath filter against the NETCONF stream. An "establish-subscription" RPC to meet this objective might be:

```
POST /restconf/operations/ietf-subscribed-notifications:establish-subscription
{
   "ietf-subscribed-notifications:input": {
      "stream": "NETCONF",
      "stream-xpath-filter": "/ietf-vrrp:vrrp-protocol-error-event[protocol-erro
r-reason='checksum-error']/",
   }
}
```

Figure 16: Establishing a subscription error reason via XPath

For more examples of XPath filters, see [XPATH].

Suppose the "establish-subscription" in Figure 16 was accepted. And suppose later a subscriber decided they wanted to broaden this subscription cover to all VRRP protocol events (i.e., not just those with a "checksum error"). The subscriber might attempt to modify the subscription in a way which replaces the XPath filter with a subtree filter which sends all VRRP protocol events to a subscriber. Such a "modify-subscription" RPC might look like:

```
POST /restconf/operations/ietf-subscribed-notifications:modify-subscription
{
   "ietf-subscribed-notifications:input": {
      "stream": "NETCONF",
      "stream-subtree-filter": {
        "/ietf-vrrp:vrrp-protocol-error-event" : {}
      }
   }
}
```

Figure 17

For more examples of subtree filters, see [RFC6241], section 6.4.

Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

v08 - v09

- o Addressed comments received during WGLC.

v07 - v08

- o Aligned with RESTCONF mechanism.
- o YANG model: removed augment of subscription-started, added restconf transport.

o Tweaked Appendix A.1 to match draft-ietf-netconf-netconf-event-notifications-13.

o Added Appendix A.3 for filter example.

v06 - v07

o Removed configured subscriptions.

o Subscription identifier renamed to id.

v05 - v06

o JSON examples updated by Reshad.

v04 - v05

o Error mechanisms updated to match embedded RESTCONF mechanisms

o Restructured format and sections of document.

o Added a YANG data model for HTTP specific parameters.

o Mirrored the examples from the NETCONF transport draft to allow easy comparison.

v03 - v04

o Draft not fully synched to new version of subscribed-notifications yet.

o References updated

v02 - v03

o Event notification reframed to notification message.

o Tweaks to wording/capitalization/format.

v01 - v02

o Removed sections now redundant with [I-D.draft-ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push] such as: mechanisms for subscription maintenance, terminology definitions, stream discovery.

o 3rd party subscriptions are out-of-scope.

- o SSE only used with RESTCONF and HTTP1.1 dynamic subscriptions
- o Timeframes for event tagging are self-defined.
- o Clean-up of wording, references to terminology, section numbers.

v00 - v01

- o Removed the ability for more than one subscription to go to a single HTTP2 stream.
- o Updated call flows. Extensively.
- o SSE only used with RESTCONF and HTTP1.1 dynamic subscriptions
- o HTTP is not used to determine that a receiver has gone silent and is not Receiving Event Notifications
- o Many clean-ups of wording and terminology

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Reshad Rahman
Cisco Systems

Email: rrahman@cisco.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

K. Watsen
Juniper Networks
G. Wu
Cisco Systems
L. Xia
Huawei

October 22, 2018

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-08

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:

o "2018-10-22" --> the publication date of this draft

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

o Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Table of Contents

1. Introduction   3
2. Terminology   4
3. The SSH Client Model   4
  3.1. Tree Diagram   4
  3.2. Example Usage   5
  3.3. YANG Module   8
4. The SSH Server Model 12
  4.1. Tree Diagram 12

4.2. Example Usage 12
4.3. YANG Module 16
5. The SSH Common Model 20
5.1. Tree Diagram 22
5.2. Example Usage 23
5.3. YANG Module 23
6. Security Considerations 33
7. IANA Considerations 34
7.1. The IETF XML Registry 34
7.2. The YANG Module Names Registry 34
8. References 35
8.1. Normative References 35
8.2. Informative References 36
Appendix A. Change Log 38
A.1. 00 to 01 38
A.2. 01 to 02 38
A.3. 02 to 03 38
A.4. 03 to 04 38
A.5. 04 to 05 39
A.6. 05 to 06 39
A.7. 06 to 07 39
A.8. 07 to 08 39
Acknowledgements 39
Authors' Addresses 40

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

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

2. Terminology

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

3. The SSH Client Model

3.1. Tree Diagram

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

```
module: ietf-ssh-client

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

3.2. Example Usage

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

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

[Note: '\' line wrapping for formatting only]

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

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

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

  <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-client>
```

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

[Note: '\' line wrapping for formatting only]

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

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

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

  <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-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@2018-10-22.yang"
module ietf-ssh-client {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
  prefix "sshc";

  import ietf-ssh-common {
    prefix sshcmn;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
  }

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

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

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

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

    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>

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


  description
   "This module defines a reusable grouping for a SSH client that
```

```
      can be used as a basis for specific SSH client instances.

      Copyright (c) 2018 IETF Trust and the persons identified as
      authors of the code. All rights reserved.

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

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

    revision "2018-10-22" {
      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.";
    }

    // groupings

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

    grouping client-identity-grouping {
      description
        "A reusable grouping for configuring a SSH client identity.";
      container client-identity {
        description
          "The credentials used by the client to authenticate to
```

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

    grouping server-auth-grouping {
      description
        "A reusable grouping for configuring SSH server
         authentication.";
      container server-auth {
        must 'pinned-ssh-host-keys or pinned-ca-certs or '
             + 'pinned-server-certs';
        description
          "Trusted server identities.";
```

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

        leaf pinned-server-certs {
          if-feature sshcmn:ssh-x509-certs;
          if-feature "ta:x509-certificates";
          type ta:pinned-certificates-ref;
          description
            "A reference to a list of server certificates used by
             the SSH client to authenticate SSH server certificates.
             A server certificate is authenticated if it is an
             exact match to a configured server certificate.";
          reference
            "RFC YYYY: YANG Data Model for Global Trust Anchors";
        }
      } // end server-auth
    } // end server-auth-grouping

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

```
        uses sshcmn:transport-params-grouping;
      }
    } // end transport-params-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

  grouping transport-params-grouping
    +-- transport-params {ssh-server-transport-params-config}?
       +---u transport-params-grouping
  grouping client-auth-grouping
    +-- client-cert-auth {sshcmn:ssh-x509-certs}?
       +-- pinned-ca-certs?       ta:pinned-certificates-ref
       |       {ta:x509-certificates}?
       +-- pinned-client-certs?   ta:pinned-certificates-ref
               {ta:x509-certificates}?
  grouping server-identity-grouping
    +-- server-identity
       +-- host-key* [name]
          +-- name?                 string
          +-- (host-key-type)
             +--:(public-key)
             |  +-- public-key
             |     +---u server-identity-grouping
             +--:(certificate)
                +-- certificate {sshcmn:ssh-x509-certs}?
                   +---u server-identity-grouping
  grouping ssh-server-grouping
    +---u server-identity-grouping
    +---u client-auth-grouping
    +---u transport-params-grouping
```

4.2. Example Usage

This section presents two examples showing the ssh-server-grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 3 of

[I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

[Note: '\' line wrapping for formatting only]

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

  <!-- which host-keys will this SSH server present -->
  <server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto\
-types">ct:rsa2048</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
      </public-key>
    </host-key>
  </server-identity>

  <!-- which client-certs will this SSH server trust -->
  <client-cert-auth>
    <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
erts>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
ent-certs>
  </client-cert-auth>

  <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>
```

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

[Note: '\' line wrapping for formatting only]

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

  <!-- which host-keys will this SSH server present -->
  <server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <reference>ex-rsa-key</reference>
      </public-key>
    </host-key>
  </server-identity>

  <!-- which client-certs will this SSH server trust -->
  <client-cert-auth>
    <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
erts>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
ent-certs>
  </client-cert-auth>

  <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@2018-10-22.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 2018-10-22; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
  }

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

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

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

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

    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>

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


  description
```

```
       "This module defines a reusable grouping for a SSH server that
        can be used as a basis for specific SSH server instances.

        Copyright (c) 2018 IETF Trust and the persons identified as
        authors of the code. All rights reserved.

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

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

      revision "2018-10-22" {
        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.";
      }

      // groupings

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

      grouping server-identity-grouping {
        description
          "A reusable grouping for configuring an SSH server identity.";
        container server-identity {
          description
```

```
          "The list of host-keys the SSH server will present when
           establishing a SSH connection.";
        list host-key {
          key name;
          min-elements 1;
          ordered-by user;
          description
            "An ordered list of host keys the SSH server will use to
             construct its ordered list of algorithms, when sending
             its SSH_MSG_KEXINIT message, as defined in Section 7.1
             of RFC 4253.";
          reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer
                       Protocol";
          leaf name {
            type string;
            description
              "An arbitrary name for this host-key";
          }
          choice host-key-type {
            mandatory true;
            description
              "The type of host key being specified";
            container public-key {
              uses ks:local-or-keystore-asymmetric-key-grouping;
              description
                "A locally-defined or referenced asymmetric key pair
                 to be used for the SSH server's host key.";
              reference
                "RFC ZZZZ: YANG Data Model for a Centralized
                           Keystore Mechanism";
            }
            container certificate {
              if-feature sshcmn:ssh-x509-certs;
              uses
                ks:local-or-keystore-end-entity-cert-with-key-grouping;
              description
                "A locally-defined or referenced end-entity
                 certificate to be used for the SSH server's
                 host key.";
              reference
                "RFC ZZZZ: YANG Data Model for a Centralized
                           Keystore Mechanism";
            }
          }
        }
      } // end server-identity
    } // end server-identity-grouping
```

```
     grouping client-auth-grouping {
       description
         "A reusable grouping for configuring a SSH client
          authentication.";
       container client-cert-auth {
         if-feature sshcmn:ssh-x509-certs;
         description
           "A reference to a list of pinned certificate authority (CA)
            certificates and a reference to a list of pinned client
            certificates.

            Note: password and public-key based client authentication
                  are not configured in this YANG module as they are
                  expected to be configured by the ietf-system module
                  defined in RFC 7317.";
         reference
           "RFC 7317: A YANG Data Model for System Management";
         leaf pinned-ca-certs {
           if-feature "ta:x509-certificates";
           type ta:pinned-certificates-ref;
           description
             "A reference to a list of certificate authority (CA)
              certificates used by the SSH server to authenticate
              SSH client certificates.  A client certificate is
              authenticated if it has a valid chain of trust to
              a configured pinned CA certificate.";
           reference
             "RFC YYYY: YANG Data Model for Global Trust Anchors";
         }
         leaf pinned-client-certs {
           if-feature "ta:x509-certificates";
           type ta:pinned-certificates-ref;
           description
             "A reference to a list of client certificates used by
              the SSH server to authenticate SSH client certificates.
              A clients certificate is authenticated if it is an
              exact match to a configured pinned client certificate.";
           reference
             "RFC YYYY: YANG Data Model for Global Trust Anchors";
         }
       }
     } // end client-auth-grouping

     grouping transport-params-grouping {
       description
         "A reusable grouping for configuring a SSH transport
          parameters.";
       container transport-params {
```

```
         if-feature ssh-server-transport-params-config;
         description
           "Configurable parameters of the SSH transport layer.";
         uses sshcmn:transport-params-grouping;
       }
     } // end transport-params-grouping

   }
   <CODE ENDS>
```

5. The SSH Common Model

The SSH common model presented in this section contains identities and groupings common to both SSH clients and SSH servers. The transport-params-grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the SSH transport layer connection. The ability to restrict the the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

[I-D.ietf-netconf-crypto-types] defines six categories of cryptographic algorithms (hash-algorithm, symmetric-key-encryption-algorithm, mac-algorithm, asymmetric-key-encryption-algorithm, signature-algorithm, key-negotiation-algorithm) and lists several widely accepted algorithms for each of them. The SSH client and server models use one or more of these algorithms. The SSH common model includes four parameters for configuring its permitted SSH algorithms, which are: host-key-alg, key-exchange-alg, encryption-alg and mac-alg. The following tables are provided, in part, to define the subset of algorithms defined in the crypto-types model used by SSH and, in part, to ensure compatibility of configured SSH cryptographic parameters for configuring its permitted SSH algorithms ("sshcmn" representing SSH common model, and "ct" representing crypto-types model which the SSH client/server model is based on):

sshcmn:host-key-alg	ct:signature-algorithm
dsa-sha1	dsa-sha1
rsa-pkcs1-sha1	rsa-pkcs1-sha1
rsa-pkcs1-sha256	rsa-pkcs1-sha256
rsa-pkcs1-sha512	rsa-pkcs1-sha512
ecdsa-secp256r1-sha256	ecdsa-secp256r1-sha256
ecdsa-secp384r1-sha384	ecdsa-secp384r1-sha384
ecdsa-secp521r1-sha512	ecdsa-secp521r1-sha512
x509v3-rsa-pkcs1-sha1	x509v3-rsa-pkcs1-sha1
x509v3-rsa2048-pkcs1-sha256	x509v3-rsa2048-pkcs1-sha1
x509v3-ecdsa-secp256r1-sha256	x509v3-ecdsa-secp256r1-sha256
x509v3-ecdsa-secp384r1-sha384	x509v3-ecdsa-secp384r1-sha384
x509v3-ecdsa-secp521r1-sha512	x509v3-ecdsa-secp521r1-sha512

Table 1 The SSH Host-key-alg Compatibility Matrix

sshcmn:key-exchange-alg	ct:key-negotiation-algorithm
diffie-hellman-group14-sha1	diffie-hellman-group14-sha1
diffie-hellman-group14-sha256	diffie-hellman-group14-sha256
diffie-hellman-group15-sha512	diffie-hellman-group15-sha512
diffie-hellman-group16-sha512	diffie-hellman-group16-sha512
diffie-hellman-group17-sha512	diffie-hellman-group17-sha512
diffie-hellman-group18-sha512	diffie-hellman-group18-sha512
ecdh-sha2-secp256r1	ecdh-sha2-secp256r1
ecdh-sha2-secp384r1	ecdh-sha2-secp384r1

Table 2 The SSH Key-exchange-alg Compatibility Matrix

sshcmn:encryption-alg	ct:symmetric-key-encryption-algorithm
aes-128-cbc	aes-128-cbc
aes-192-cbc	aes-192-cbc
aes-256-cbc	aes-256-cbc
aes-128-ctr	aes-128-ctr
aes-192-ctr	aes-192-ctr
aes-256-ctr	aes-256-ctr

Table 3 The SSH Encryption-alg Compatibility Matrix

sshcmn:mac-alg	ct:mac-algorithm
hmac-sha1	hmac-sha1
hmac-sha1-96	hmac-sha1-96
hmac-sha2-256	hmac-sha2-256
hmac-sha2-512	hmac-sha2-512

Table 4 The SSH Mac-alg Compatibility Matrix

As is seen in the tables above, the names of the "sshcmn" algorithms are all identical to the names of algorithms defined in [I-D.ietf-netconf-crypto-types]. While appearing to be redundant, it is important to realize that not all the algorithms defined in [I-D.ietf-netconf-crypto-types] are supported by SSH. That is, the algorithms supported by SSH are a subset of the algorithms defined in [I-D.ietf-netconf-crypto-types]. The algorithms used by SSH are redefined in this document in order to constrain the algorithms that may be selected to just the ones used by SSH.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive. As well, some algorithms that are REQUIRED by [RFC4253] are missing, notably "ssh-dss" and "diffie-hellman-group1-sha1" due to their weak security and there being alternatives that are widely supported.

5.1. Tree Diagram

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

```
module: ietf-ssh-common

  grouping transport-params-grouping
    +-- host-key
    |  +-- host-key-alg*   identityref
    +-- key-exchange
    |  +-- key-exchange-alg*   identityref
    +-- encryption
    |  +-- encryption-alg*   identityref
    +-- mac
       +-- mac-alg*   identityref
```

5.2. Example Usage

This following example illustrates how the transport-params-grouping appears when populated with some data.

```
<transport-params
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common"
  xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">
  <host-key>
    <host-key-alg>algs:x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>algs:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      algs:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>algs:aes256-ctr</encryption-alg>
    <encryption-alg>algs:aes192-ctr</encryption-alg>
    <encryption-alg>algs:aes128-ctr</encryption-alg>
    <encryption-alg>algs:aes256-cbc</encryption-alg>
    <encryption-alg>algs:aes192-cbc</encryption-alg>
    <encryption-alg>algs:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>algs:hmac-sha2-256</mac-alg>
    <mac-alg>algs:hmac-sha2-512</mac-alg>
  </mac>
</transport-params>
```

5.3. YANG Module

This YANG module has normative references to [RFC4253], [RFC4344], [RFC4419], [RFC5656], [RFC6187], and [RFC6668].

```
<CODE BEGINS> file "ietf-ssh-common@2018-10-22.yang"
module ietf-ssh-common {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix "sshcmn";

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

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

```
        WG List:  <mailto:netconf@ietf.org>

        Author:   Kent Watsen
                  <mailto:kwatsen@juniper.net>

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

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

       Copyright (c) 2018 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
     }

     // features

     feature ssh-ecc {
       description
         "Elliptic Curve Cryptography is supported for SSH.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
                    Secure Shell Transport Layer";
     }

     feature ssh-x509-certs {
       description
         "X.509v3 certificates are supported for SSH per RFC 6187.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
```

```
     }

     feature ssh-dh-group-exchange {
       description
         "Diffie-Hellman Group Exchange is supported for SSH.";
       reference
         "RFC 4419: Diffie-Hellman Group Exchange for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     feature ssh-ctr {
       description
         "SDCTR encryption mode is supported for SSH.";
       reference
         "RFC 4344: The Secure Shell (SSH) Transport Layer
                    Encryption Modes";
     }

     feature ssh-sha2 {
       description
         "The SHA2 family of cryptographic hash functions is
          supported for SSH.";
       reference
         "FIPS PUB 180-4: Secure Hash Standard (SHS)";
     }

     // identities

     identity public-key-alg-base {
       description
         "Base identity used to identify public key algorithms.";
     }

     identity ssh-dss {
       base public-key-alg-base;
       description
         "Digital Signature Algorithm using SHA-1 as the
          hashing algorithm.";
       reference
         "RFC 4253:
            The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity ssh-rsa {
       base public-key-alg-base;
       description
         "RSASSA-PKCS1-v1_5 signature scheme using SHA-1 as the
          hashing algorithm.";
```

```
        reference
          "RFC 4253:
             The Secure Shell (SSH) Transport Layer Protocol";
      }

      identity ecdsa-sha2-nistp256 {
        base public-key-alg-base;
        if-feature "ssh-ecc and ssh-sha2";
        description
          "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
           nistp256 curve and the SHA2 family of hashing algorithms.";
        reference
          "RFC 5656: Elliptic Curve Algorithm Integration in the
                     Secure Shell Transport Layer";
      }

      identity ecdsa-sha2-nistp384 {
        base public-key-alg-base;
        if-feature "ssh-ecc and ssh-sha2";
        description
          "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
           nistp384 curve and the SHA2 family of hashing algorithms.";
        reference
          "RFC 5656: Elliptic Curve Algorithm Integration in the
                     Secure Shell Transport Layer";
      }

      identity ecdsa-sha2-nistp521 {
        base public-key-alg-base;
        if-feature "ssh-ecc and ssh-sha2";
        description
          "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
           nistp521 curve and the SHA2 family of hashing algorithms.";
        reference
          "RFC 5656: Elliptic Curve Algorithm Integration in the
                     Secure Shell Transport Layer";
      }

      identity x509v3-ssh-rsa {
        base public-key-alg-base;
        if-feature ssh-x509-certs;
        description
          "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
           in an X.509v3 certificate and using SHA-1 as the hashing
           algorithm.";
        reference
          "RFC 6187: X.509v3 Certificates for Secure Shell
                     Authentication";
```

```
     }

     identity x509v3-rsa2048-sha256 {
       base public-key-alg-base;
       if-feature "ssh-x509-certs and ssh-sha2";
       description
         "RSASSA-PKCS1-v1_5 signature scheme using a public key stored
          in an X.509v3 certificate and using SHA-256 as the hashing
          algorithm.  RSA keys conveyed using this format MUST have a
          modulus of at least 2048 bits.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity x509v3-ecdsa-sha2-nistp256 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA)
          using the nistp256 curve with a public key stored in
          an X.509v3 certificate and using the SHA2 family of
          hashing algorithms.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity x509v3-ecdsa-sha2-nistp384 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA)
          using the nistp384 curve with a public key stored in
          an X.509v3 certificate and using the SHA2 family of
          hashing algorithms.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity x509v3-ecdsa-sha2-nistp521 {
       base public-key-alg-base;
       if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
       description
         "Elliptic Curve Digital Signature Algorithm (ECDSA)
          using the nistp521 curve with a public key stored in
          an X.509v3 certificate and using the SHA2 family of
```

```
          hashing algorithms.";
       reference
         "RFC 6187: X.509v3 Certificates for Secure Shell
                    Authentication";
     }

     identity key-exchange-alg-base {
       description
         "Base identity used to identify key exchange algorithms.";
     }

     identity diffie-hellman-group14-sha1 {
       base key-exchange-alg-base;
       description
         "Diffie-Hellman key exchange with SHA-1 as HASH and
          Oakley Group 14 (2048-bit MODP Group).";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity diffie-hellman-group-exchange-sha1 {
       base key-exchange-alg-base;
       if-feature ssh-dh-group-exchange;
       description
         "Diffie-Hellman Group and Key Exchange with SHA-1 as HASH.";
       reference
         "RFC 4419: Diffie-Hellman Group Exchange for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     identity diffie-hellman-group-exchange-sha256 {
       base key-exchange-alg-base;
       if-feature "ssh-dh-group-exchange and ssh-sha2";
       description
         "Diffie-Hellman Group and Key Exchange with SHA-256 as HASH.";
       reference
         "RFC 4419: Diffie-Hellman Group Exchange for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     identity ecdh-sha2-nistp256 {
       base key-exchange-alg-base;
       if-feature "ssh-ecc and ssh-sha2";
       description
         "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
          nistp256 curve and the SHA2 family of hashing algorithms.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
```

```
                    Secure Shell Transport Layer";
     }

     identity ecdh-sha2-nistp384 {
       base key-exchange-alg-base;
       if-feature "ssh-ecc and ssh-sha2";
       description
         "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
          nistp384 curve and the SHA2 family of hashing algorithms.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
                    Secure Shell Transport Layer";
     }

     identity ecdh-sha2-nistp521 {
       base key-exchange-alg-base;
       if-feature "ssh-ecc and ssh-sha2";
       description
         "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
          nistp521 curve and the SHA2 family of hashing algorithms.";
       reference
         "RFC 5656: Elliptic Curve Algorithm Integration in the
                    Secure Shell Transport Layer";
     }

     identity encryption-alg-base {
       description
         "Base identity used to identify encryption algorithms.";
     }

     identity triple-des-cbc {
       base encryption-alg-base;
       description
         "Three-key 3DES in CBC mode.";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity aes128-cbc {
       base encryption-alg-base;
       description
        "AES in CBC mode, with a 128-bit key.";
       reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity aes192-cbc {
       base encryption-alg-base;
```

```
        description
          "AES in CBC mode, with a 192-bit key.";
        reference
          "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      }

      identity aes256-cbc {
        base encryption-alg-base;
        description
          "AES in CBC mode, with a 256-bit key.";
        reference
          "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      }

      identity aes128-ctr {
        base encryption-alg-base;
        if-feature ssh-ctr;
        description
          "AES in SDCTR mode, with 128-bit key.";
        reference
          "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
                     Modes";
      }

      identity aes192-ctr {
        base encryption-alg-base;
        if-feature ssh-ctr;
        description
          "AES in SDCTR mode, with 192-bit key.";
        reference
          "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
                     Modes";
      }

      identity aes256-ctr {
        base encryption-alg-base;
        if-feature ssh-ctr;
        description
          "AES in SDCTR mode, with 256-bit key.";
        reference
          "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
             Modes";
      }

      identity mac-alg-base {
        description
          "Base identity used to identify message authentication
           code (MAC) algorithms.";
```

```
     }

     identity hmac-sha1 {
       base mac-alg-base;
       description
         "HMAC-SHA1";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

     identity hmac-sha2-256 {
       base mac-alg-base;
       if-feature "ssh-sha2";
       description
         "HMAC-SHA2-256";
       reference
         "RFC 6668: SHA-2 Data Integrity Verification for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     identity hmac-sha2-512 {
       base mac-alg-base;
       if-feature "ssh-sha2";
       description
         "HMAC-SHA2-512";
       reference
         "RFC 6668: SHA-2 Data Integrity Verification for the
                    Secure Shell (SSH) Transport Layer Protocol";
     }

     // groupings

     grouping transport-params-grouping {
       description
         "A reusable grouping for SSH transport parameters.";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
       container host-key {
         description
           "Parameters regarding host key.";
         leaf-list host-key-alg {
           type identityref {
             base public-key-alg-base;
           }
           ordered-by user;
           description
             "Acceptable host key algorithms in order of descending
              preference.  The configured host key algorithms should
```

```
              be compatible with the algorithm used by the configured
              private key.  Please see Section 5 of RFC XXXX for
              valid combinations.

              If this leaf-list is not configured (has zero elements)
              the acceptable host key algorithms are implementation-
              defined.";
           reference
             "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
         }
       }
       container key-exchange {
         description
           "Parameters regarding key exchange.";
         leaf-list key-exchange-alg {
           type identityref {
             base key-exchange-alg-base;
           }
           ordered-by user;
           description
            "Acceptable key exchange algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable key exchange algorithms are implementation
             defined.";
         }
       }
       container encryption {
         description
           "Parameters regarding encryption.";
         leaf-list encryption-alg {
           type identityref {
             base encryption-alg-base;
           }
           ordered-by user;
           description
            "Acceptable encryption algorithms in order of descending
             preference.

             If this leaf-list is not configured (has zero elements)
             the acceptable encryption algorithms are implementation
             defined.";
         }
       }
       container mac {
         description
           "Parameters regarding message authentication code (MAC).";
```

```
         leaf-list mac-alg {
           type identityref {
             base mac-alg-base;
           }
           ordered-by user;
           description
             "Acceptable MAC algorithms in order of descending
              preference.

              If this leaf-list is not configured (has zero elements)
              the acceptable MAC algorithms are implementation-
              defined.";
         }
       }

     } // transport-params-grouping

   }
   <CODE ENDS>
```

6. Security Considerations

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

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

Since the modules defined in this document define only groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

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

/client-auth/password: This node in the 'ietf-ssh-client' module is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. The only time this node should be returned is to support backup/ restore type workflows. However, no NACM annotations are applied as the data SHOULD be writable by users other than a designated 'recovery session'.

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

NONE

7. IANA Considerations

7.1. The IETF XML Registry

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

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

7.2. The YANG Module Names Registry

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

```
name:         ietf-ssh-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-client
prefix:       sshc
reference:    RFC XXXX
name:         ietf-ssh-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-server
prefix:       sshs
reference:    RFC XXXX
name:         ietf-ssh-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-common
prefix:       sshcmn
reference:    RFC XXXX
```

8. References

8.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-01 (work in progress), September 2018.

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-06 (work in progress), September 2018.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "YANG Data Model for Global Trust Anchors", draft-ietf-netconf-trust-anchors-01 (work in progress), September 2018.

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

[RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <https://www.rfc-editor.org/info/rfc4344>.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <https://www.rfc-editor.org/info/rfc4419>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

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

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <https://www.rfc-editor.org/info/rfc6668>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

8.2. Informative References

[OPENSSH] "OpenSSH", 2016, <http://www.openssh.com>.

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

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

[RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, DOI 10.17487/RFC4254, January 2006, <https://www.rfc-editor.org/info/rfc4254>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

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

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Change Log

A.1. 00 to 01

- Noted that '0.0.0.0' and '::' might have special meanings.

- Renamed "keychain" to "keystore".

A.2. 01 to 02

- Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.

- Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.

- Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

A.3. 02 to 03

- Removed 'RESTRICTED' enum from 'password' leaf type.

- 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

- Change title to "YANG Groupings for SSH Clients and SSH Servers"

- Added reference to RFC 6668

- Added RFC 8174 to Requirements Language Section.

- Enhanced description statement for ietf-ssh-server's "trusted-ca-certs" leaf.

- Added mandatory true to ietf-ssh-client's "client-auth" 'choice' statement.

- Changed the YANG prefix for module ietf-ssh-common from 'sshcom' to 'sshcmn'.

- Removed the compression algorithms as they are not commonly configurable in vendors' implementations.

- Updating descriptions in transport-params-grouping and the servers'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 matricies to Section 5 (thanks Frank!)

- Claified that any configured "host-key-alg" values need to be compatible with the configured private key.

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
Juniper Networks

EMail: kwatsen@juniper.net

Gary Wu
Cisco Systems

EMail: garywu@cisco.com

Liang Xia
Huawei

EMail: frank.xialiang@huawei.com

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2019

E. Voit
Cisco Systems
A. Clemm
Huawei
A. Gonzalez Prieto
Microsoft
E. Nilsen-Nygaard
A. Tripathy
Cisco Systems
October 23, 2018

Subscription to YANG Event Notifications
draft-ietf-netconf-subscribed-notifications-18

Abstract

This document defines a YANG data model and associated mechanisms enabling subscriber-specific subscriptions to a publisher's event streams. Applying these elements allows a subscriber to request for and receive a continuous, custom feed of publisher generated information.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Motivation 3
1.2. Terminology 3
1.3. Solution Overview 5
1.4. Relationship to RFC 5277 6
2. Solution 6
2.1. Event Streams 7
2.2. Event Stream Filters 7
2.3. QoS 8
2.4. Dynamic Subscriptions 9
2.5. Configured Subscriptions 17
2.6. Event Record Delivery 24
2.7. subscription state change notifications 25
2.8. Subscription Monitoring 31
2.9. Advertisement 32
3. YANG Data Model Trees 32
3.1. Event Streams Container 32
3.2. Filters Container 33
3.3. Subscriptions Container 33
4. Data Model 35
5. Considerations 62
5.1. IANA Considerations 62
5.2. Implementation Considerations 62
5.3. Transport Requirements 63
5.4. Security Considerations 64
6. Acknowledgments 68
7. References 68
7.1. Normative References 68
7.2. Informative References 70
Appendix A. Example Configured Transport Augmentation 70
Appendix B. Changes between revisions 72
Authors' Addresses 77

1. Introduction

This document defines a YANG data model and associated mechanisms enabling subscriber-specific subscriptions to a publisher's event streams. Effectively this enables a 'subscribe then publish' capability where the customized information needs and access permissions of each target receiver are understood by the publisher before subscribed event records are marshaled and pushed. The receiver then gets a continuous, custom feed of publisher generated information.

While the functionality defined in this document is transport-agnostic, transports like NETCONF [RFC6241] or RESTCONF [RFC8040] can be used to configure or dynamically signal subscriptions, and there are bindings defined for subscribed event record delivery for NETCONF within [I-D.draft-ietf-netconf-netconf-event-notifications], and for HTTP2 or HTTP1.1 within [I-D.draft-ietf-netconf-restconf-notif].

The YANG model in this document conforms to the Network Management Datastore Architecture defined in [RFC8342].

1.1. Motivation

Various limitations in [RFC5277] are discussed in [RFC7923]. Resolving these issues is the primary motivation for this work. Key capabilities supported by this document include:

- multiple subscriptions on a single transport session
- support for dynamic and configured subscriptions
- modification of an existing subscription in progress
- per-subscription operational counters
- negotiation of subscription parameters (through the use of hints returned as part of declined subscription requests)
- subscription state change notifications (e.g., publisher driven suspension, parameter modification)
- independence from transport

1.2. 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.

Client: defined in [RFC8342].

Configuration: defined in [RFC8342].

Configuration datastore: defined in [RFC8342].

Configured subscription: A subscription installed via configuration into a configuration datastore.

Dynamic subscription: A subscription created dynamically by a subscriber via a remote procedure call.

Event: An occurrence of something that may be of interest. Examples include a configuration change, a fault, a change in status, crossing a threshold, or an external input to the system.

Event occurrence time: a timestamp matching the time an originating process identified as when an event happened.

Event record: A set of information detailing an event.

Event stream: A continuous, chronologically ordered set of events aggregated under some context.

Event stream filter: Evaluation criteria which may be applied against event records within an event stream. Event records pass the filter when specified criteria are met.

Notification message: Information intended for a receiver indicating that one or more events have occurred.

Publisher: An entity responsible for streaming notification messages per the terms of a subscription.

Receiver: A target to which a publisher pushes subscribed event records. For dynamic subscriptions, the receiver and subscriber are the same entity.

Subscriber: A client able to request and negotiate a contract for the generation and push of event records from a publisher. For dynamic subscriptions, the receiver and subscriber are the same entity.

Subscription: A contract with a publisher, stipulating which information one or more receivers wish to have pushed from the publisher without the need for further solicitation.

All YANG tree diagrams used in this document follow the notation defined in [RFC8340].

1.3. Solution Overview

This document describes a transport agnostic mechanism for subscribing to and receiving content from an event stream within a publisher. This mechanism is through the use of a subscription.

Two types of subscriptions are supported:

1. Dynamic subscriptions, where a subscriber initiates a subscription negotiation with a publisher via an RPC. If the publisher is able to serve this request, it accepts it, and then starts pushing notification messages back to the subscriber. If the publisher is not able to serve it as requested, then an error response is returned. This response MAY include hints at subscription parameters that, had they been present, may have enabled the dynamic subscription request to be accepted.

2. Configured subscriptions, which allow the management of subscriptions via a configuration so that a publisher can send notification messages to a receiver. Support for configured subscriptions is optional, with its availability advertised via a YANG feature.

Additional characteristics differentiating configured from dynamic subscriptions include:

- The lifetime of a dynamic subscription is bound by the transport session used to establish it. For connection-oriented stateful transports like NETCONF, the loss of the transport session will result in the immediate termination of any associated dynamic subscriptions. For connectionless or stateless transports like HTTP, a lack of receipt acknowledgment of a sequential set of notification messages and/or keep-alives can be used to trigger a termination of a dynamic subscription. Contrast this to the lifetime of a configured subscription. This lifetime is driven by relevant configuration being present within the publisher's applied configuration. Being tied to configuration operations implies configured subscriptions can be configured to persist across reboots, and implies a configured subscription can persist even when its publisher is fully disconnected from any network.

- Configured subscriptions can be modified by any configuration client with write permission on the configuration of the subscription. Dynamic subscriptions can only be modified via an RPC request made by the original subscriber, or a change to configuration data referenced by the subscription.

Note that there is no mixing-and-matching of dynamic and configured operations on a single subscription. Specifically, a configured subscription cannot be modified or deleted using RPCs defined in this document. Also note that transport specific transport drafts based on this specification MUST detail the life cycles of both dynamic and configured subscriptions.

A publisher MAY terminate a dynamic subscription at any time. Similarly, it MAY decide to temporarily suspend the sending of notification messages for any dynamic subscription, or for one or more receivers of a configured subscription. Such termination or suspension is driven by internal considerations of the publisher.

1.4. Relationship to RFC 5277

This document is intended to provide a superset of the subscription capabilities initially defined within [RFC5277]. Especially when extending an existing [RFC5277] implementation, it is important to understand what has been reused and what has been replaced. Key relationships between these two documents include:

- o this document defines a transport independent capability, [RFC5277] is specific to NETCONF.
- o the data model in this document is used instead of the data model in Section 3.4 of [RFC5277] for the new operations.
- o the RPC operations in this draft replace the operation "create-subscription" defined in [RFC5277], section 4.
- o the <notification> message of [RFC5277], Section 4 is used.
- o the included contents of the "NETCONF" event stream are identical between this document and [RFC5277].
- o a publisher MAY implement both the Notification Management Schema and RPCs defined in [RFC5277] and this new document concurrently.
- o unlike [RFC5277], this document enables a single transport session to intermix notification messages and RPCs for different subscriptions.

2. Solution

Per the overview provided in Section 1.3, this section details the overall context, state machines, and subsystems which may be assembled to allow the subscription of events from a publisher.

2.1. Event Streams

An event stream is a named entity on a publisher which exposes a continuously updating set of YANG encoded event records. An event record is an intantiation of a "notification" YANG statement. If the "notification" is defined as a child to a data node, the intantiation includes the hierarchy of nodes that identifies the data node in the datastore (see Section 7.16.2 of [RFC7950]). Each event stream is available for subscription. It is out of the scope of this document to identify a) how event streams are defined (other than the NETCONF stream), b) how event records are defined/generated, and c) how event records are assigned to event streams.

There is only one reserved event stream name within this document: "NETCONF". The "NETCONF" event stream contains all NETCONF event record information supported by the publisher, except where an event record has explicitly been excluded from the stream. Beyond the "NETCONF" stream, implementations MAY define additional event streams.

As YANG encoded event records are created by a system, they may be assigned to one or more streams. The event record is distributed to a subscription's receiver(s) where: (1) a subscription includes the identified stream, and (2) subscription filtering does not exclude the event record from that receiver.

Access control permissions may be used to silently exclude event records from within an event stream for which the receiver has no read access. As an example of how this might be accomplished, see [RFC8341] section 3.4.6. Note that per Section 2.7 of this document, subscription state change notifications are never filtered out.

If no access control permissions are in place for event records on an event stream, then a receiver MUST be allowed access to all the event records. If subscriber permissions change during the lifecycle of a subscription and event stream access is no longer permitted, then the subscription MUST be terminated.

Event records MUST NOT be delivered to a receiver in a different order than they were placed onto an event stream.

2.2. Event Stream Filters

This document defines an extensible filtering mechanism. The filter itself is a boolean test which is placed on the content of an event record. A 'false' filtering result causes the event message to be excluded from delivery to a receiver. A filter never results in information being stripped from within an event record prior to that event record being encapsulated within a notification message. The two optional event stream filtering syntaxes supported are [XPATH] and subtree [RFC6241].

If no event stream filter is provided within a subscription, all event records on an event stream are to be sent.

2.3. QoS

This document provide for several QoS parameters. These parameters indicate the treatment of a subscription relative to other traffic between publisher and receiver. Included are:

- A "dscp" marking to differentiate prioritization of notification messages during network transit.
- A "weighting" so that bandwidth proportional to this weighting can be allocated to this subscription relative to other subscriptions.
- a "dependency" upon another subscription.

If the publisher supports the "dscp" feature, then a subscription with a "dscp" leaf MUST result in a corresponding [RFC2474] DSCP marking being placed within the IP header of any resulting notification messages and subscription state change notifications.

For the "weighting" parameter, when concurrently dequeuing notification messages from multiple subscriptions to a receiver, the publisher MUST allocate bandwidth to each subscription proportionally to the weights assigned to those subscriptions. "Weighting" is an optional capability of the publisher; support for it is identified via the "qos" feature.

If a subscription has the "dependency" parameter set, then any buffered notification messages containing event records selected by the parent subscription MUST be dequeued prior to the notification messages of the dependent subscription. If notification messages have dependencies on each other, the notification message queued the longest MUST go first. If a "dependency" included within an RPC references a subscription which does not exist or is no longer accessible to that subscriber, that "dependency" MUST be silently removed. "Dependency" is an optional capability of the publisher; support for it is identified via the "qos" feature.

2.4. Dynamic Subscriptions

Dynamic subscriptions are managed via protocol operations (in the form of [RFC7950], Section 7.14 RPCs) made against targets located within the publisher. These RPCs have been designed extensibly so that they may be augmented for subscription targets beyond event streams. For examples of such augmentations, see the RPC augmentations within [I-D.ietf-netconf-yang-push]'s YANG model.

2.4.1. Dynamic Subscription State Model

Below is the publisher's state machine for a dynamic subscription. Each state is shown in its own box. It is important to note that such a subscription doesn't exist at the publisher until an "establish-subscription" RPC is accepted. The mere request by a subscriber to establish a subscription is insufficient for that subscription to be externally visible. Start and end states are depicted to reflect subscription creation and deletion events.

```
                  .........
                  : start :
                  :.......:
                      |
             establish-subscription
                      |
                      |   .-------modify-subscription--------.
                      v   v                                  |
                .-----------.                          .-----------.
     .--------. | receiver  |--insufficient CPU, b/w-->| receiver  |
 modify-       '|  active   |                          | suspended |
 subscription   |           |<----CPU, b/w sufficient--|           |
     ---------->'-----------'                          '-----------'
                      |                                      |
           delete/kill-subscription                     delete/kill-
                      |                                 subscription
                      v                                      |
                  .........                                  |
                  :  end  :<---------------------------------'
                  :.......:
```

Figure 1: Publisher's state for a dynamic subscription

Of interest in this state machine are the following:

o Successful "establish-subscription" or "modify-subscription" RPCs put the subscription into the active state.

- o Failed "modify-subscription" RPCs will leave the subscription in its previous state, with no visible change to any streaming updates.

- o A "delete-subscription" or "kill-subscription" RPC will end the subscription, as will the reaching of a "stop-time".

- o A publisher may choose to suspend a subscription when there is insufficient CPU or bandwidth available to service the subscription. This is notified to a subscriber with a "subscription-suspended" subscription state change notification.

- o A suspended subscription may be modified by the subscriber (for example in an attempt to use fewer resources). Successful modification returns the subscription to the active state.

- o Even without a "modify-subscription" request, a publisher may return a subscription to the active state should the resource constraints become sufficient again. This is announced to the subscriber via the "subscription-resumed" subscription state change notification.

2.4.2. Establishing a Dynamic Subscription

The "establish-subscription" RPC allows a subscriber to request the creation of a subscription.

The input parameters of the operation are:

- o A "stream" name which identifies the targeted event stream against which the subscription is applied.

- o An event stream filter which may reduce the set of event records pushed.

- o Where the transport used by the RPC supports multiple encodings, an optional "encoding" for the event records pushed. If no "encoding" is included, the encoding of the RPC MUST be used.

- o An optional "stop-time" for the subscription. If no "stop-time" is present, notification messages will continue to be sent until the subscription is terminated.

- o An optional "replay-start-time" for the subscription. The "replay-start-time" MUST be in the past and indicates that the subscription is requesting a replay of previously generated information from the event stream. For more on replay, see

Section 2.4.2.1. Where there is no "replay-start-time", the subscription starts immediately.

If the publisher can satisfy the "establish-subscription" request, it replies with an identifier for the subscription, and then immediately starts streaming notification messages.

Below is a tree diagram for "establish-subscription". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---x establish-subscription
   +---w input
   |  +---w (target)
   |  |  +--:(stream)
   |  |     +---w (stream-filter)?
   |  |     |  +--:(by-reference)
   |  |     |  |  +---w stream-filter-name
   |  |     |  |          stream-filter-ref
   |  |     |  +--:(within-subscription)
   |  |     |     +---w (filter-spec)?
   |  |     |        +--:(stream-subtree-filter)
   |  |     |        |  +---w stream-subtree-filter?   <anydata>
   |  |     |        |          {subtree}?
   |  |     |        +--:(stream-xpath-filter)
   |  |     |           +---w stream-xpath-filter?
   |  |     |                   yang:xpath1.0 {xpath}?
   |  |     +---w stream                               stream-ref
   |  |     +---w replay-start-time?
   |  |             yang:date-and-time {replay}?
   |  +---w stop-time?
   |  |       yang:date-and-time
   |  +---w dscp?                                      inet:dscp
   |  |       {dscp}?
   |  +---w weighting?                                 uint8
   |  |       {qos}?
   |  +---w dependency?
   |  |       subscription-id {qos}?
   |  +---w encoding?                                  encoding
   +--ro output
      +--ro id                             subscription-id
      +--ro replay-start-time-revision?   yang:date-and-time
              {replay}?
```

Figure 2: establish-subscription RPC tree diagram

A publisher MAY reject the "establish-subscription" RPC for many reasons as described in Section 2.4.6. The contents of the resulting RPC error response MAY include details on input parameters which if considered in a subsequent "establish-subscription" RPC, may result in a successful subscription establishment. Any such hints MUST be transported within a yang-data "establish-subscription-stream-error-info" container included within the RPC error response.

```
    yang-data establish-subscription-stream-error-info
       +--ro establish-subscription-stream-error-info
          +--ro reason?                  identityref
          +--ro filter-failure-hint?     string
```

Figure 3: establish-subscription RPC yang-data tree diagram

2.4.2.1. Requesting a replay of event records

Replay provides the ability to establish a subscription which is also capable of passing recently generated event records. In other words, as the subscription initializes itself, it sends any event records within the target event stream which meet the filter criteria, which have an event time which is after the "replay-start-time", and which have an event time before the "stop-time" should this "stop-time" exist. The end of these historical event records is identified via a "replay-completed" subscription state change notification. Any event records generated since the subscription establishment may then follow. For a particular subscription, all event records will be delivered in the order they are placed into the event stream.

Replay is an optional feature which is dependent on an event stream supporting some form of logging. This document puts no restrictions on the size or form of the log, where it resides within the publisher, or when event record entries in the log are purged.

The inclusion of a "replay-start-time" within an "establish-subscription" RPC indicates a replay request. If the "replay-start-time" contains a value that is earlier than what a publisher's retained history supports, then if the subscription is accepted, the actual publisher's revised start time MUST be set in the returned "replay-start-time-revision" object.

A "stop-time" parameter may be included in a replay subscription. For a replay subscription, the "stop-time" MAY be earlier than the current time, but MUST be later than the "replay-start-time".

If the given "replay-start-time" is later than the time marked within any event records retained within the replay buffer, then the publisher MUST send a "replay-completed" notification immediately after a successful establish-subscription RPC response.

If an event stream supports replay, the "replay-support" leaf is present in the "/streams/stream" list entry for the event stream. An event stream that does support replay is not expected to have an unlimited supply of saved notifications available to accommodate any given replay request. To assess the timeframe available for replay, subscribers can read the leafs "replay-log-creation-time" and "replay-log-aged-time". See Figure 18 for the YANG tree, and Section 4 for the YANG model describing these elements. The actual size of the replay log at any given time is a publisher specific matter. Control parameters for the replay log are outside the scope of this document.

2.4.3. Modifying a Dynamic Subscription

The "modify-subscription" operation permits changing the terms of an existing dynamic subscription. Dynamic subscriptions can be modified any number of times. Dynamic subscriptions can only be modified via this RPC using a transport session connecting to the subscriber. If the publisher accepts the requested modifications, it acknowledges success to the subscriber, then immediately starts sending event records based on the new terms.

Subscriptions created by configuration cannot be modified via this RPC. However configuration may be used to modify objects referenced by the subscription (such as a referenced filter).

Below is a tree diagram for "modify-subscription". All objects contained in this tree are described within the included YANG model within Section 4.

```
      +---x modify-subscription
         +---w input
            +---w id
            |       subscription-id
            +---w (target)
            |  +--:(stream)
            |     +---w (stream-filter)?
            |        +--:(by-reference)
            |        |  +---w stream-filter-name
            |        |          stream-filter-ref
            |        +--:(within-subscription)
            |           +---w (filter-spec)?
            |              +--:(stream-subtree-filter)
            |              |  +---w stream-subtree-filter?   <anydata>
            |              |          {subtree}?
            |              +--:(stream-xpath-filter)
            |                 +---w stream-xpath-filter?
            |                         yang:xpath1.0 {xpath}?
            +---w stop-time?
                    yang:date-and-time
```

Figure 4: modify-subscription RPC tree diagram

If the publisher accepts the requested modifications on a currently suspended subscription, the subscription will immediately be resumed (i.e., the modified subscription is returned to the active state.) The publisher MAY immediately suspend this newly modified subscription through the "subscription-suspended" notification before any event records are sent.

If the publisher rejects the RPC request, the subscription remains as prior to the request. That is, the request has no impact whatsoever. Rejection of the RPC for any reason is indicated by via RPC error as described in Section 2.4.6. The contents of such a rejected RPC MAY include hints on inputs which (if considered) may result in a successfully modified subscription. These hints MUST be transported within a yang-data "modify-subscription-stream-error-info" container inserted into the RPC error response.

Below is a tree diagram for "modify-subscription-RPC-yang-data". All objects contained in this tree are described within the included YANG model within Section 4.

```
yang-data modify-subscription-stream-error-info
   +--ro modify-subscription-stream-error-info
      +--ro reason?                identityref
      +--ro filter-failure-hint?   string
```

Figure 5: modify-subscription RPC yang-data tree diagram

2.4.4. Deleting a Dynamic Subscription

The "delete-subscription" operation permits canceling an existing subscription. If the publisher accepts the request, and the publisher has indicated success, the publisher MUST NOT send any more notification messages for this subscription. If the delete request matches a known subscription established on the same transport session, then it MUST be deleted; otherwise it MUST be rejected with no changes to the publisher.

Below is a tree diagram for "delete-subscription". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---x delete-subscription
   +---w input
      +---w id     subscription-id
```

Figure 6: delete-subscription RPC tree diagram

Dynamic subscriptions can only be deleted via this RPC using a transport session connecting to the subscriber. Configured subscriptions cannot be deleted using RPCs.

2.4.5. Killing a Dynamic Subscription

The "kill-subscription" operation permits an operator to end a dynamic subscription which is not associated with the transport session used for the RPC. A publisher MUST terminate any dynamic subscription identified by the "id" parameter in the RPC request, if such a subscription exists.

Configured subscriptions cannot be killed using this RPC. Instead, configured subscriptions are deleted as part of regular configuration operations. Publishers MUST reject any RPC attempt to kill a configured subscription.

Below is a tree diagram for "kill-subscription". All objects contained in this tree are described within the included YANG model within Section 4.

```
     +---x kill-subscription
       +---w input
          +---w id    subscription-id
```

Figure 7: kill-subscription RPC tree diagram

2.4.6. RPC Failures

Whenever an RPC is unsuccessful, the publisher returns relevant information as part of the RPC error response. Transport level error processing MUST be done before RPC error processing described in this section. In all cases, RPC error information returned will use existing transport layer RPC structures, such as those seen with NETCONF in [RFC6241] Appendix A, or with RESTCONF in [RFC8040] Section 7.1. These structures MUST be able to encode subscription specific errors identified below and defined within this document's YANG model.

As a result of this mixture, how subscription errors are encoded within an RPC error response is transport dependent. Following are valid errors which can occur for each RPC:

```
 establish-subscription        modify-subscription
 ----------------------        -------------------
 dscp-unavailable              filter-unsupported
 encoding-unsupported          insufficient-resources
 filter-unsupported            no-such-subscription
 insufficient-resources
 replay-unsupported

 delete-subscription           kill-subscription
 ----------------------        ----------------------
 no-such-subscription           no-such-subscription
```

To see a NETCONF based example of an error response from above, see [I-D.draft-ietf-netconf-netconf-event-notifications], Figure 10.

There is one final set of transport independent RPC error elements included in the YANG model. These are three yang-data structures which enable the publisher to provide to the receiver that error information which does not fit into existing transport layer RPC structures. These three yang-data structures are:

1. "establish-subscription-stream-error-info": This MUST be returned with the leaf "reason" populated if an RPC error reason has not been placed elsewhere within the transport portion of a failed

"establish-subscription" RPC response. This MUST be sent if hints on how to overcome the RPC error are included.

2. "modify-subscription-stream-error-info": This MUST be returned with the leaf "reason" populated if an RPC error reason has not been placed elsewhere within the transport portion of a failed "modify-subscription" RPC response. This MUST be sent if hints on how to overcome the RPC error are included.

3. "delete-subscription-error-info": This MUST be returned with the leaf "reason" populated if an RPC error reason has not been placed elsewhere within the transport portion of a failed "delete-subscription" or "kill-subscription" RPC response.

2.5. Configured Subscriptions

A configured subscription is a subscription installed via configuration. Configured subscriptions may be modified by any configuration client with the proper permissions. Subscriptions can be modified or terminated via configuration at any point of their lifetime. Multiple configured subscriptions MUST be supportable over a single transport session.

Configured subscriptions have several characteristics distinguishing them from dynamic subscriptions:

- persistence across publisher reboots,
- persistence even when transport is unavailable, and
- an ability to send notification messages to more than one receiver (note that receivers are unaware of the existence of any other receivers.)

On the publisher, supporting configured subscriptions is optional and advertised using the "configured" feature. On a receiver of a configured subscription, support for dynamic subscriptions is optional except where replaying missed event records is required.

In addition to the subscription parameters available to dynamic subscriptions described in Section 2.4.2, the following additional parameters are also available to configured subscriptions:

- A "transport" which identifies the transport protocol to use to connect with all subscription receivers.

o One or more receivers, each intended as the destination for event records. Note that each individual receiver is identifiable by its "name".

o Optional parameters to identify where traffic should egress a publisher:

   * A "source-interface" which identifies the egress interface to use from the publisher. Publisher support for this is optional and advertised using the "interface-designation" feature.

   * A "source-address" address, which identifies the IP address to stamp on notification messages destined for the receiver.

   * A "source-vrf" which identifies the VRF on which to reach receivers. This VRF is a network instance as defined within [I-D.draft-ietf-rtgwg-ni-model]. Publisher support for VRFs is optional and advertised using the "supports-vrf" feature.

   If none of the above parameters are set, notification messages MUST egress the publisher's default interface.

A tree diagram describing these parameters is shown in Figure 20 within Section 3.3. All parameters are described within the YANG model in Section 4.

2.5.1. Configured Subscription State Model

Below is the state machine for a configured subscription on the publisher. This state machine describes the three states (valid, invalid, and concluded), as well as the transitions between these states. Start and end states are depicted to reflect configured subscription creation and deletion events. The creation or modification of a configured subscription initiates an evaluation by the publisher to determine if the subscription is in valid or invalid states. The publisher uses its own criteria in making this determination. If in the valid state, the subscription becomes operational. See (1) in the diagram below.

```
 .........
 : start :-.
 :.......: |
      create  .---modify-----.----------------------------------.
           |  |              |                                  |
           V  V          .-------.         .......         .---------.
  .----[evaluate]--no--->|invalid|-delete->: end :<-delete-|concluded|
  |                      '-------'         :.....:         '---------'
  |-[evaluate]--no-(2).      ^                ^                 ^
  |        ^          |      |                |                 |
 yes       |          '->unsupportable      delete          stop-time
  |      modify         (subscription-  (subscription-  (subscription-
  |        |             terminated*)    terminated*)     concluded*)
  |        |                 |                |                 |
 (1)       |                (3)              (4)               (5)
  |   .---------------------------------------------------------------.
  '-->|                         valid                                 |
      '---------------------------------------------------------------'

 Legend:
 dotted boxes: subscription added or removed via configuration
 dashed boxes: states for a subscription
 [evaluate]: decision point on whether the subscription is supportable
 (*): resulting subscription state change notification
```

Figure 8: Publisher state model for a configured subscription

A subscription in the valid state may move to the invalid state in one of two ways. First, it may be modified in a way which fails a re-evaluation. See (2) in the diagram. Second, the publisher might determine that the subscription is no longer supportable. This could be for reasons of an unexpected but sustained increase in an event stream's event records, degraded CPU capacity, a more complex referenced filter, or other higher priority subscriptions which have usurped resources. See (3) in the diagram. No matter the case, a "subscription-terminated" notification is sent to any receivers in an active or suspended state. A subscription in the valid state may also transition to the concluded state via (5) if a configured stop time has been reached. In this case, a "subscription-concluded" notification is sent to any receivers in active or suspended states. Finally, a subscription may be deleted by configuration (4).

When a subscription is in the valid state, a publisher will attempt to connect with all receivers of a configured subscription and deliver notification messages. Below is the state machine for each receiver of a configured subscription. This receiver state machine is fully contained within the state machine of the configured subscription, and is only relevant when the configured subscription is in the valid state.

```
     .-----------------------------------------------------------------.
     |                         valid                                   |
     |   .----------.                           .------------.         |
     |   | receiver |---timeout---------------->|  receiver  |         |
     |   |connecting|<----------------reset--(c)|disconnected|         |
     |   |          |<-transport                '------------'         |
     |   '----------'  loss,reset------------------------------.       |
     |      (a)          |                                     |       |
     |  subscription-   (b)                                   (b)      |
     |  started*    .--------.                             .---------. |
     |       '----->|        |(d)-insufficient CPU,------->|         | |
     |              |receiver|    buffer overflow          |receiver | |
     | subscription-| active |                             |suspended| |
     |   modified*  |        |<----CPU, b/w sufficient,-(e)|         | |
     |        '---->'--------'     subscription-modified*  '---------' |
     '-----------------------------------------------------------------'
```

Legend:
dashed boxes which include the word 'receiver' show the possible states for an individual receiver of a valid configured subscription.
* indicates a subscription state change notification

Figure 9: Receiver state for a configured subscription on a Publisher

When a configured subscription first moves to the valid state, the "state" leaf of each receiver is initialized to the connecting state. If transport connectivity is not available to any receiver and there are any notification messages to deliver, a transport session is established (e.g., through [RFC8071]). Individual receivers are moved to the active state when a "subscription-started" subscription state change notification is successfully passed to that receiver (a). Event records are only sent to active receivers. Receivers of a configured subscription remain active if both transport connectivity can be verified to the receiver, and event records are not being dropped due to a publisher buffer overflow. The result is that a receiver will remain active on the publisher as long as events aren't being lost, or the receiver cannot be reached. In addition, a configured subscription's receiver MUST be moved to the connecting state if the receiver is reset via the "reset" action (b), (c). For more on reset, see Section 2.5.5. If transport connectivity cannot be achieved while in the connecting state, the receiver MAY be moved to the disconnected state.

A configured subscription's receiver MUST be moved to the suspended state if there is transport connectivity between the publisher and receiver, but notification messages are failing to be delivered due to publisher buffer overflow, or notification messages are not able to be generated for that receiver due to insufficient CPU (d). This is indicated to the receiver by the "subscription-suspended" subscription state change notification.

A configured subscription receiver MUST be returned to the active state from the suspended state when notification messages are able to be generated, bandwidth is sufficient to handle the notification messages, and a receiver has successfully been sent a "subscription-resumed" or "subscription-modified" subscription state change notification (e). The choice as to which of these two subscription state change notifications is sent is determined by whether the subscription was modified during the period of suspension.

Modification of a configured subscription is possible at any time. A "subscription-modified" subscription state change notification will be sent to all active receivers, immediately followed by notification messages conforming to the new parameters. Suspended receivers will also be informed of the modification. However this notification will await the end of the suspension for that receiver (e).

The mechanisms described above are mirrored in the RPCs and notifications within the document. It should be noted that these RPCs and notifications have been designed to be extensible and allow subscriptions into targets other than event streams. For instance, the YANG module defined in Section 5 of [I-D.ietf-netconf-yang-push] augments "/sn:modify-subscription/sn:input/sn:target".

2.5.2. Creating a Configured Subscription

Configured subscriptions are established using configuration operations against the top-level "subscriptions" subtree.

Because there is no explicit association with an existing transport session, configuration operations MUST include additional parameters beyond those of dynamic subscriptions. These parameters identify each receiver, how to connect with that receiver, and possibly whether the notification messages need to come from a specific egress interface on the publisher. Receiver specific transport connectivity parameters MUST be configured via transport specific augmentations to this specification. See Section 2.5.7 for details.

After a subscription is successfully established, the publisher immediately sends a "subscription-started" subscription state change notification to each receiver. It is quite possible that upon configuration, reboot, or even steady-state operations, a transport session may not be currently available to the receiver. In this case, when there is something to transport for an active subscription, transport specific call-home operations will be used to establish the connection. When transport connectivity is available, notification messages may then be pushed.

With active configured subscriptions, it is allowable to buffer event records even after a "subscription-started" has been sent. However if events are lost (rather than just delayed) due to replay buffer overflow, a new "subscription-started" must be sent. This new "subscription-started" indicates an event record discontinuity.

To see an example of subscription creation using configuration operations over NETCONF, see Appendix A of [I-D.draft-ietf-netconf-netconf-event-notifications].

2.5.3. Modifying a Configured Subscription

Configured subscriptions can be modified using configuration operations against the top-level "subscriptions" subtree.

If the modification involves adding receivers, added receivers are placed in the connecting state. If a receiver is removed, the subscription state change notification "subscription-terminated" is sent to that receiver if that receiver is active or suspended.

If the modification involves changing the policies for the subscription, the publisher sends to currently active receivers a "subscription-modified" notification. For any suspended receivers, a "subscription-modified" notification will be delayed until the receiver is resumed. (Note: in this case, the "subscription-modified" notification informs the receiver that the subscription has been resumed, so no additional "subscription-resumed" need be sent. Also note that if multiple modifications have occurred during the suspension, only the "subscription-modified" notification describing the latest one need be sent to the receiver.)

2.5.4. Deleting a Configured Subscription

Subscriptions can be deleted through configuration against the top-level "subscriptions" subtree.

Immediately after a subscription is successfully deleted, the publisher sends to all receivers of that subscription a subscription state change notification stating the subscription has ended (i.e., "subscription-terminated").

2.5.5. Resetting a Configured Subscription Receiver

It is possible that a configured subscription to a receiver needs to be reset. This is accomplished via the "reset" action within the YANG model at "/subscriptions/subscription/receivers/receiver/reset". This action may be useful in cases where a publisher has timed out trying to reach a receiver. When such a reset occurs, a transport session will be initiated if necessary, and a new "subscription-started" notification will be sent. This action does not have any effect on transport connectivity if the needed connectivity already exists.

2.5.6. Replay for a Configured Subscription

It is possible to do replay on a configured subscription. This is supported via the configuration of the "configured-replay" object on the subscription. The setting of this object enables the streaming of the buffered event records for the subscribed event stream. All buffered event records which have been retained since the last publisher restart will be sent to each configured receiver.

Replay of events records created since restart is useful. It allows event records generated before transport connectivity establishment to be passed to a receiver. Setting the restart time as the earliest configured replay time precludes possibility of resending of event records logged prior to publisher restart. It also ensures the same records will be sent to each configured receiver, regardless of the speed of transport connectivity establishment to each receiver. Finally, establishing restart as the earliest potential time for event records to be included within notification messages, a well-understood timeframe for replay is defined.

As a result, when any configured subscription receivers become active, buffered event records will be sent immediately after the "subscription-started" notification. If the publisher knows the last event record sent to a receiver, and the publisher has not rebooted, the next event record on the event stream which meets filtering criteria will be the leading event record sent. Otherwise, the leading event record will be the first event record meeting filtering criteria subsequent to the latest of three different times: the "replay-log-creation-time", "replay-log-aged-time", or the most recent publisher boot time. The "replay-log-creation-time" and "replay-log-aged-time" are discussed in Section 2.4.2.1. The most recent publisher boot time ensures that duplicate event records are not replayed from a previous time the publisher was booted.

It is quite possible that a receiver might want to retrieve event records from an event stream prior to the latest boot. If such records exist where there is a configured replay, the publisher MUST send the time of the event record immediately preceding the "replay-start-time" within the "replay-previous-event-time" leaf. Through the existence of the "replay-previous-event-time", the receiver will know that earlier events prior to reboot exist. In addition, if the subscriber was previously receiving event records with the same subscription "id", the receiver can determine if there was a timegap where records generated on the publisher were not successully received. And with this information, the receiver may choose to dynamically subscribe to retrieve any event records placed into the event stream before the most recent boot time.

All other replay functionality remains the same as with dynamic subscriptions as described in Section 2.4.2.1.

2.5.7. Transport Connectivity for a Configured Subscription

This specification is transport independent. However supporting a configured subscription will often require the establishment of transport connectivity. And the parameters used for this transport connectivity establishment are transport specific. As a result, the YANG model defined within Section 4 is not able to directly define and expose these transport parameters.

It is necessary for an implementation to support the connection establishment process. To support this function, the YANG model does include a node where transport specific parameters for a particular receiver may be augmented. This node is "/subscriptions/subscription/receivers/receiver". By augmenting transport parameters from this node, system developers are able to incorporate the YANG objects necessary to support the transport connectivity establishment process.

The result of this is the following requirement. A publisher supporting the feature "configured" MUST also support least one YANG model which augments transport connectivity parameters on "/subscriptions/subscription/receivers/receiver". For an example of such an augmentation, see Appendix A.

2.6. Event Record Delivery

Whether dynamic or configured, once a subscription has been set up, the publisher streams event records via notification messages per the terms of the subscription. For dynamic subscriptions, notification messages are sent over the session used to establish the subscription. For configured subscriptions, notification messages are sent over the connections specified by the transport and each receiver of a configured subscription.

A notification message is sent to a receiver when an event record is not blocked by either the specified filter criteria or receiver permissions. This notification message MUST include an "eventTime" object as defined per [RFC5277] Section 4. This "eventTime" MUST be at the top level of YANG structured event record.

The following example within [RFC7950] section 7.16.3 is an example of a compliant message:

```
<notification
      xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2007-09-01T10:00:00Z</eventTime>
    <link-failure xmlns="http://acme.example.com/system">
        <if-name>so-1/2/3.0</if-name>
        <if-admin-status>up</if-admin-status>
        <if-oper-status>down</if-oper-status>
    </link-failure>
</notification>
```

Figure 10: subscribed notification message

When a dynamic subscription has been started or modified, with "establish-subscription" or "modify-subscription" respectively, event records matching the newly applied filter criteria MUST NOT be sent until after the RPC reply has been sent.

When a configured subscription has been started or modified, event records matching the newly applied filter criteria MUST NOT be sent until after the "subscription-started" or "subscription-modified" notifications has been sent, respectively.

2.7. subscription state change notifications

In addition to sending event records to receivers, a publisher MUST also send subscription state change notifications when events related to subscription management have occurred.

subscription state change notifications are unlike other notifications in that they are never included in any event stream. Instead, they are inserted (as defined in this section) within the sequence of notification messages sent to a particular receiver. subscription state change notifications cannot be filtered out, they cannot be stored in replay buffers, and they are delivered only to impacted receivers of a subscription. The identification of subscription state change notifications is easy to separate from other notification messages through the use of the YANG extension "subscription-state-notif". This extension tags a notification as a subscription state change notification.

The complete set of subscription state change notifications is described in the following subsections.

2.7.1. subscription-started

This notification indicates that a configured subscription has started, and event records may be sent. Included in this subscription state change notification are all the parameters of the subscription, except for the receiver(s) transport connection information and origin information indicating where notification messages will egress the publisher. Note that if a referenced filter from the "filters" container has been used within the subscription, the notification still provides the contents of that referenced filter under the "within-subscription" subtree.

Note that for dynamic subscriptions, no "subscription-started" notifications are ever sent.

Below is a tree diagram for "subscription-started". All objects contained in this tree are described within the included YANG model within Section 4.

```
       +---n subscription-started {configured}?
          +--ro id
          |       subscription-id
          +--ro (target)
          |  +--:(stream)
          |     +--ro (stream-filter)?
          |     |  +--:(by-reference)
          |     |  |  +--ro stream-filter-name
          |     |  |          stream-filter-ref
          |     |  +--:(within-subscription)
          |     |     +--ro (filter-spec)?
          |     |        +--:(stream-subtree-filter)
          |     |        |  +--ro stream-subtree-filter?   <anydata>
          |     |        |          {subtree}?
          |     |        +--:(stream-xpath-filter)
          |     |           +--ro stream-xpath-filter?     yang:xpath1.0
          |     |                   {xpath}?
          |     +--ro stream                               stream-ref
          |     +--ro replay-start-time?
          |     |       yang:date-and-time {replay}?
          |     +--ro replay-previous-event-time?
          |             yang:date-and-time {replay}?
          +--ro stop-time?
          |       yang:date-and-time
          +--ro dscp?                                      inet:dscp
          |       {dscp}?
          +--ro weighting?                                 uint8 {qos}?
          +--ro dependency?
          |       subscription-id {qos}?
          +--ro transport?                                 transport
          |       {configured}?
          +--ro encoding?                                  encoding
          +--ro purpose?                                   string
                  {configured}?
```

Figure 11: subscription-started notification tree diagram

2.7.2. subscription-modified

This notification indicates that a subscription has been modified by configuration operations. It is delivered directly after the last event records processed using the previous subscription parameters, and before any event records processed after the modification.

Below is a tree diagram for "subscription-modified". All objects contained in this tree are described within the included YANG model within Section 4.

```
      +---n subscription-modified
         +--ro id
         |       subscription-id
         +--ro (target)
         |  +--:(stream)
         |     +--ro (stream-filter)?
         |     |  +--:(by-reference)
         |     |  |  +--ro stream-filter-name
         |     |  |          stream-filter-ref
         |     |  +--:(within-subscription)
         |     |     +--ro (filter-spec)?
         |     |        +--:(stream-subtree-filter)
         |     |        |  +--ro stream-subtree-filter?   <anydata>
         |     |        |          {subtree}?
         |     |        +--:(stream-xpath-filter)
         |     |           +--ro stream-xpath-filter?     yang:xpath1.0
         |     |                   {xpath}?
         |     +--ro stream                               stream-ref
         |     +--ro replay-start-time?
         |             yang:date-and-time {replay}?
         +--ro stop-time?
         |       yang:date-and-time
         +--ro dscp?                                      inet:dscp
         |       {dscp}?
         +--ro weighting?                                 uint8 {qos}?
         +--ro dependency?
         |       subscription-id {qos}?
         +--ro transport?                                 transport
         |       {configured}?
         +--ro encoding?                                  encoding
         +--ro purpose?                                   string
                 {configured}?
```

Figure 12: subscription-modified notification tree diagram

A publisher most often sends this notification directly after the modification of any configuration parameters impacting a configured subscription. But it may also be sent at two other times:

1. Where a configured subscription has been modified during the suspension of a receiver, the notification will be delayed until the receiver's suspension is lifted. In this situation, the notification indicates that the subscription has been both modified and resumed.

2. A "subscription-modified" subscription state change notification MUST be sent if the contents of the filter identified by the subscription's "stream-filter-ref" leaf has changed. This state

change notification is to be sent for a filter change impacting any active receiver of a configured or dynamic subscription.

2.7.3. subscription-terminated

This notification indicates that no further event records for this subscription should be expected from the publisher. A publisher may terminate the sending event records to a receiver for the following reasons:

1. Configuration which removes a configured subscription, or a "kill-subscription" RPC which ends a dynamic subscription. These are identified via the reason "no-such-subscription".

2. A referenced filter is no longer accessible. This is identified by "filter-unavailable".

3. The event stream referenced by a subscription is no longer accessible by the receiver. This is identified by "stream-unavailable".

4. A suspended subscription has exceeded some timeout. This is identified by "suspension-timeout".

Each of the reasons above correspond one-to-one with a "reason" identityref specified within the YANG model.

Below is a tree diagram for "subscription-terminated". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---n subscription-terminated
   +--ro id        subscription-id
   +--ro reason    identityref
```

Figure 13: subscription-terminated notification tree diagram

Note: this subscription state change notification MUST be sent to a dynamic subscription’s receiver when the subscription ends unexpectedly. The cases when this might happen are when a "kill-subscription" RPC is successful, or when some other event not including the reaching the subscription’s "stop-time" results in a publisher choosing to end the subscription.

2.7.4. subscription-suspended

This notification indicates that a publisher has suspended the sending of event records to a receiver, and also indicates the possible loss of events. Suspension happens when capacity constraints stop a publisher from serving a valid subscription. The two conditions where is this possible are:

1. "insufficient-resources" when a publisher is unable to produce the requested event stream of notification messages, and

2. "unsupportable-volume" when the bandwidth needed to get generated notification messages to a receiver exceeds a threshold.

These conditions are encoded within the "reason" object. No further notification will be sent until the subscription resumes or is terminated.

Below is a tree diagram for "subscription-suspended". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---n subscription-suspended
   +--ro id        subscription-id
   +--ro reason    identityref
```

Figure 14: subscription-suspended notification tree diagram

2.7.5. subscription-resumed

This notification indicates that a previously suspended subscription has been resumed under the unmodified terms previously in place. Subscribed event records generated after the issuance of this subscription state change notification may now be sent.

Below is the tree diagram for "subscription-resumed". All objects contained in this tree are described within the included YANG model within Section 4.

```
+---n subscription-resumed
   +--ro id    subscription-id
```

Figure 15: subscription-resumed notification tree diagram

2.7.6. subscription-completed

This notification indicates that a subscription that includes a "stop-time" has successfully finished passing event records upon the reaching of that time.

Below is a tree diagram for "subscription-completed". All objects contained in this tree are described within the included YANG model within Section 4.

```
    +---n subscription-completed {configured}?
       +--ro id    subscription-id
```

Figure 16: subscription-completed notification tree diagram

2.7.7. replay-completed

This notification indicates that all of the event records prior to the current time have been passed to a receiver. It is sent before any notification message containing an event record with a timestamp later than (1) the "stop-time" or (2) the subscription's start time.

If a subscription contains no "stop-time", or has a "stop-time" that has not been reached, then after the "replay-completed" notification has been sent, additional event records will be sent in sequence as they arise naturally on the publisher.

Below is a tree diagram for "replay-completed". All objects contained in this tree are described within the included YANG model within Section 4.

```
    +---n replay-completed {replay}?
       +--ro id    subscription-id
```

Figure 17: replay-completed notification tree diagram

2.8. Subscription Monitoring

In the operational state datastore, the container "subscriptions" maintains the state of all dynamic subscriptions, as well as all configured subscriptions. Using datastore retrieval operations, or subscribing to the "subscriptions" container [I-D.ietf-netconf-yang-push] allows the state of subscriptions and their connectivity to receivers to be monitored.

Each subscription in the operational state datastore is represented as a list element. Included in this list are event counters for each receiver, the state of each receiver, as well as the subscription parameters currently in effect. The appearance of the leaf "configured-subscription-state" indicates that a particular subscription came into being via configuration. This leaf also indicates if the current state of that subscription is valid, invalid, and concluded.

To understand the flow of event records within a subscription, there are two counters available for each receiver. The first counter is "sent-event-records" which shows the quantity of events actually identified for sending to a receiver. The second counter is "excluded-event-records" which shows event records not sent to receiver. "excluded-event-records" shows the combined results of both access control and per-subscription filtering. For configured subscriptions, counters are reset whenever the subscription is evaluated to valid (see (1) in Figure 8).

Dynamic subscriptions are removed from the operational state datastore once they expire (reaching stop-time) or when they are terminated. While many subscription objects are shown as configurable, dynamic subscriptions are only included within the operational state datastore and as a result are not configurable.

2.9. Advertisement

Publishers supporting this document MUST indicate support of the YANG model "ietf-subscribed-notifications" within the YANG library of the publisher. In addition if supported, the optional features "encode-xml", "encode-json", "configured" "supports-vrf", "qos", "xpath", "subtree", "interface-designation", "dscp", and "replay" MUST be indicated.

3. YANG Data Model Trees

This section contains tree diagrams for nodes defined in Section 4. For tree diagrams of subscription state change notifications, see Section 2.7. For the tree diagrams for the RPCs, see Section 2.4.

3.1. Event Streams Container

A publisher maintains a list of available event streams as operational data. This list contains both standardized and vendor-specific event streams. This enables subscribers to discover what streams a publisher supports.

```
     +--ro streams
        +--ro stream* [name]
           +--ro name                         string
           +--ro description                  string
           +--ro replay-support?              empty {replay}?
           +--ro replay-log-creation-time    yang:date-and-time
           |       {replay}?
           +--ro replay-log-aged-time?       yang:date-and-time
                   {replay}?
```

Figure 18: Stream Container tree diagram

Above is a tree diagram for the "streams" container. All objects contained in this tree are described within the included YANG model within Section 4.

3.2. Filters Container

The "filters" container maintains a list of all subscription filters that persist outside the life-cycle of a single subscription. This enables pre-defined filters which may be referenced by more than one subscription.

```
     +--rw filters
        +--rw stream-filter* [name]
           +--rw name                           string
           +--rw (filter-spec)?
              +--:(stream-subtree-filter)
              |  +--rw stream-subtree-filter?   <anydata> {subtree}?
              +--:(stream-xpath-filter)
                 +--rw stream-xpath-filter?     yang:xpath1.0 {xpath}?
```

Figure 19: Filter Container tree diagram

Above is a tree diagram for the filters container. All objects contained in this tree are described within the included YANG model within Section 4.

3.3. Subscriptions Container

The "subscriptions" container maintains a list of all subscriptions on a publisher, both configured and dynamic. It can be used to retrieve information about the subscriptions which a publisher is serving.

```
     +--rw subscriptions
```

```
        +--rw subscription* [id]
           +--rw id
           |       subscription-id
           +--rw (target)
           |  +--:(stream)
           |     +--rw (stream-filter)?
           |     |  +--:(by-reference)
           |     |  |  +--rw stream-filter-name
           |     |  |          stream-filter-ref
           |     |  +--:(within-subscription)
           |     |     +--rw (filter-spec)?
           |     |        +--:(stream-subtree-filter)
           |     |        |  +--rw stream-subtree-filter?   <anydata>
           |     |        |          {subtree}?
           |     |        +--:(stream-xpath-filter)
           |     |           +--rw stream-xpath-filter?
           |     |                   yang:xpath1.0 {xpath}?
           |     +--rw stream                               stream-ref
           |     +--ro replay-start-time?
           |     |       yang:date-and-time {replay}?
           |     +--rw configured-replay?                   empty
           |             {configured,replay}?
           +--rw stop-time?
           |       yang:date-and-time
           +--rw dscp?                                      inet:dscp
           |       {dscp}?
           +--rw weighting?                                 uint8 {qos}?
           +--rw dependency?
           |       subscription-id {qos}?
           +--rw transport?                                 transport
           |       {configured}?
           +--rw encoding?                                  encoding
           +--rw purpose?                                   string
           |       {configured}?
           +--rw (notification-message-origin)? {configured}?
           |  +--:(interface-originated)
           |  |  +--rw source-interface?
           |  |          if:interface-ref {interface-designation}?
           |  +--:(address-originated)
           |     +--rw source-vrf?
           |     |       -> /ni:network-instances/network-instance/name
           |     |       {supports-vrf}?
           |     +--rw source-address?
           |             inet:ip-address-no-zone
           +--ro configured-subscription-state?            enumeration
           |       {configured}?
           +--rw receivers
              +--rw receiver* [name]
```

```
                 +--rw name                      string
                 +--ro sent-event-records?
                 |       yang:zero-based-counter64
                 +--ro excluded-event-records?
                 |       yang:zero-based-counter64
                 +--ro state                     enumeration
                 +---x reset {configured}?
                    +--ro output
                       +--ro time    yang:date-and-time
```

Figure 20: Subscriptions tree diagram

Above is a tree diagram for the subscriptions container. All objects contained in this tree are described within the included YANG model within Section 4.

4. Data Model

This module imports typedefs from [RFC6991], [RFC8343], and [RFC8040], and it references [I-D.draft-ietf-rtgwg-ni-model], [XPATH], [RFC6241], [RFC7540], [RFC7951] and [RFC7950].

[note to the RFC Editor - please replace XXXX within this YANG model with the number of this document, and XXXY with the number of [I-D.draft-ietf-rtgwg-ni-model]]

[note to the RFC Editor - please replace the two dates within the YANG module with the date of publication]

```
<CODE BEGINS> file "ietf-subscribed-notifications@2018-10-11.yang"
module ietf-subscribed-notifications {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications";

  prefix sn;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
```

```
   import ietf-netconf-acm {
     prefix nacm;
     reference
       "RFC 8341: Network Configuration Access Control Model";
   }
   import ietf-network-instance {
     prefix ni;
     reference
       "draft-ietf-rtgwg-ni-model-12: YANG Model for Network Instances";
   }
   import ietf-restconf   {
     prefix rc;
     reference
       "RFC 8040: RESTCONF Protocol";
   }
   import ietf-yang-types {
     prefix yang;
     reference
       "RFC 6991: Common YANG Data Types";
   }

   organization "IETF NETCONF (Network Configuration) Working Group";
   contact
     "WG Web:   <http:/tools.ietf.org/wg/netconf/>
      WG List:  <mailto:netconf@ietf.org>

      Author:   Alexander Clemm
                <mailto:ludwig@clemm.org>

      Author:   Eric Voit
                <mailto:evoit@cisco.com>

      Author:   Alberto Gonzalez Prieto
                <mailto:alberto.gonzalez@microsoft.com>

      Author:   Einar Nilsen-Nygaard
                <mailto:einarnn@cisco.com>

      Author:   Ambika Prasad Tripathy
                <mailto:ambtripa@cisco.com>";

   description
     "Contains a YANG specification for subscribing to event records
     and receiving matching content within notification messages.

     Copyright (c) 2018 IETF Trust and the persons identified as authors
     of the code.  All rights reserved.
```

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

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

   revision 2018-10-11 {
     description
       "Initial version";
     reference
     "RFC XXXX:Customized Subscriptions to a Publisher's Event Streams";
   }

   /*
    * FEATURES
    */

   feature configured {
     description
       "This feature indicates that configuration of subscription is
       supported.";
   }

   feature dscp {
     description
       "This feature indicates a publisher supports the placement of
       suggested prioritization levels for network transport within
       notification messages.";
   }

   feature encode-json {
     description
       "This feature indicates that JSON encoding of notification
        messages is supported.";
   }

   feature encode-xml {
     description
       "This feature indicates that XML encoding of notification
        messages is supported.";
   }

   feature interface-designation {
     description
       "This feature indicates a publisher supports sourcing all
```

```
       receiver interactions for a configured subscription from a single
       designated egress interface.";
   }

   feature qos {
     description
       "This feature indicates a publisher supports absolute
       dependencies of one subscription's traffic over another, as well
       as weighted bandwidth sharing between subscriptions.  Both of
       these are Quality of Service (QoS) features which allow
       differentiated treatment of notification messages between a
       publisher and a specific receiver.";
   }

   feature replay {
     description
       "This feature indicates that historical event record replay is
       supported.  With replay, it is possible for past event records to
       be streamed in chronological order.";
   }

   feature subtree {
     description
       "This feature indicates support for YANG subtree filtering.";
     reference "RFC 6241, Section 6.";
   }

   feature supports-vrf {
     description
       "This feature indicates a publisher supports VRF configuration
       for configured subscriptions.  VRF support for dynamic
       subscriptions does not require this feature.";
     reference "RFC XXXY, Section 6.";
   }

   feature xpath {
     description
       "This feature indicates support for XPath filtering.";
     reference "http://www.w3.org/TR/1999/REC-xpath-19991116";
   }

   /*
    * EXTENSIONS
    */

   extension subscription-state-notification {
     description
       "This statement applies only to notifications. It indicates that
```

```
        the notification is a subscription state change notification.
        Therefore it does not participate in a regular event stream and
        does not need to be specifically subscribed to in order to be
        received. This statement can only occur as a substatement to the
        YANG 'notification' statement.  This statement is not for use
        outside of this YANG module.";
   }

   /*
    * IDENTITIES
    */

   /* Identities for RPC and Notification errors */

   identity delete-subscription-error {
      description
       "Problem found while attempting to fulfill either a
       'delete-subscription' RPC request or a 'kill-subscription'
       RPC request.";
   }

   identity establish-subscription-error {
      description
       "Problem found while attempting to fulfill an
       'establish-subscription' RPC request.";
   }

   identity modify-subscription-error {
      description
       "Problem found while attempting to fulfill a
       'modify-subscription' RPC request.";
   }

   identity subscription-suspended-reason {
      description
       "Problem condition communicated to a receiver as part of a
       'subscription-terminated' notification.";
   }

   identity subscription-terminated-reason {
      description
       "Problem condition communicated to a receiver as part of a
       'subscription-terminated' notification.";
   }

   identity dscp-unavailable {
     base establish-subscription-error;
     if-feature "dscp";
```

```
     description
       "The publisher is unable mark notification messages with a
       prioritization information in a way which will be respected
       during network transit.";
   }

   identity encoding-unsupported {
     base establish-subscription-error;
     description
       "Unable to encode notification messages in the desired format.";
   }

   identity filter-unavailable {
     base subscription-terminated-reason;
     description
      "Referenced filter does not exist.  This means a receiver is
      referencing a filter which doesn't exist, or to which they do not
      have access permissions.";
   }

   identity filter-unsupported {
     base establish-subscription-error;
     base modify-subscription-error;
     description
      "Cannot parse syntax within the filter.  This failure can be from
      a syntax error, or a syntax too complex to be processed by the
      publisher.";
   }

   identity insufficient-resources {
     base establish-subscription-error;
     base modify-subscription-error;
     base subscription-suspended-reason;
     description
       "The publisher has insufficient resources to support the
        requested subscription.  An example might be that allocated CPU
        is too limited to generate the desired set of notification
        messages.";
   }

   identity no-such-subscription {
     base modify-subscription-error;
     base delete-subscription-error;
     base subscription-terminated-reason;
     description
      "Referenced subscription doesn't exist. This may be as a result of
       a non-existent subscription id, an id which belongs to another
       subscriber, or an id for configured subscription.";
```

```
   }

   identity replay-unsupported {
     base establish-subscription-error;
     if-feature "replay";
     description
      "Replay cannot be performed for this subscription. This means the
       publisher will not provide the requested historic information
       from the event stream via replay to this receiver.";
   }

   identity stream-unavailable {
     base subscription-terminated-reason;
     description
      "Not a subscribable event stream.  This means the referenced event
       stream is not available for subscription by the receiver.";
   }

   identity suspension-timeout {
     base subscription-terminated-reason;
     description
      "Termination of previously suspended subscription. The publisher
       has eliminated the subscription as it exceeded a time limit for
       suspension.";
   }

   identity unsupportable-volume {
     base subscription-suspended-reason;
     description
       "The publisher does not have the network bandwidth needed to get
       the volume of generated information intended for a receiver.";
   }

   /* Identities for encodings */

   identity configurable-encoding {
     description
       "If a transport identity derives from this identity, it means
        that it supports configurable encodings.";
   }

   identity encoding {
     description
       "Base identity to represent data encodings";
   }

   identity encode-xml {
     base encoding;
```

```
     if-feature "encode-xml";
     description
       "Encode data using XML as described in RFC 7950";
     reference
       "RFC 7950 - The YANG 1.1 Data Modeling Language";
   }

   identity encode-json {
     base encoding;
     if-feature "encode-json";
     description
       "Encode data using JSON as described in RFC 7951";
     reference
       "RFC 7951 - JSON Encoding of Data Modeled with YANG";
   }

   /* Identities for transports */
   identity transport {
     description
       "An identity that represents the underlying mechanism for
       passing notification messages.";
   }

   /*
    * TYPEDEFs
    */

   typedef encoding {
     type identityref {
       base encoding;
     }
     description
       "Specifies a data encoding, e.g. for a data subscription.";
   }

   typedef stream-filter-ref {
     type leafref {
       path "/sn:filters/sn:stream-filter/sn:name";
     }
     description
       "This type is used to reference an event stream filter.";
   }

   typedef stream-ref {
     type leafref {
       path "/sn:streams/sn:stream/sn:name";
     }
     description
```

```
       "This type is used to reference a system-provided event stream.";
   }

   typedef subscription-id {
     type uint32;
     description
       "A type for subscription identifiers.";
   }

   typedef transport {
     type identityref {
       base transport;
     }
     description
       "Specifies transport used to send notification messages to a
        receiver.";
   }

   /*
    * GROUPINGS
    */

   grouping stream-filter-elements {
     description
       "This grouping defines the base for filters applied to event
        streams.";
     choice filter-spec {
       description
         "The content filter specification for this request.";
       anydata stream-subtree-filter {
         if-feature "subtree";
         description
           "Event stream evaluation criteria encoded in the syntax of a
           subtree filter as defined in RFC 6241, Section 6.

           The subtree filter is applied to the representation of
           individual, delineated event records as contained within the
           event stream.

           If the subtree filter returns a non-empty node set, the
           filter matches the event record, and the event record is
           included in the notification message sent to the receivers.";
         reference "RFC 6241, Section 6.";
       }
       leaf stream-xpath-filter {
         if-feature "xpath";
         type yang:xpath1.0;
         description
```

```
          "Event stream evaluation criteria encoded in the syntax of
           an XPath 1.0 expression.

           The XPath expression is evaluated on the representation of
           individual, delineated event records as contained within
           the event stream.

           The result of the XPath expression is converted to a
           boolean value using the standard XPath 1.0 rules.  If the
           boolean value is 'true', the filter matches the event
           record, and the event record is included in the notification
           message sent to the receivers.

           The expression is evaluated in the following XPath context:

             o  The set of namespace declarations are those in scope on
                the 'stream-xpath-filter' leaf element.

             o  The set of variable bindings is empty.

             o  The function library is the core function library, and
                the XPath functions defined in section 10 in RFC 7950.

             o  The context node is the root node.";
        reference
          "http://www.w3.org/TR/1999/REC-xpath-19991116
           RFC 7950, Section 10.";

      }
    }
  }

  grouping update-qos {
    description
      "This grouping describes Quality of Service information
       concerning a subscription.  This information is passed to lower
       layers for transport prioritization and treatment";
    leaf dscp {
      if-feature "dscp";
      type inet:dscp;
      default "0";
      description
        "The desired network transport priority level. This is the
         priority set on notification messages encapsulating the
         results of the subscription.  This transport priority is
         shared for all receivers of a given subscription.";
    }
    leaf weighting {
```

```
       if-feature "qos";
       type uint8 {
          range "0 .. 255";
       }
       description
         "Relative weighting for a subscription. Allows an underlying
          transport layer perform informed load balance allocations
          between various subscriptions";
       reference
         "RFC-7540, section 5.3.2";
     }
     leaf dependency {
       if-feature "qos";
       type subscription-id;
       description
         "Provides the 'subscription-id' of a parent subscription which
          has absolute precedence should that parent have push updates
          ready to egress the publisher. In other words, there should be
          no streaming of objects from the current subscription if
          the parent has something ready to push.

          If a dependency is asserted via configuration or via RPC, but
          the referenced 'subscription-id' does not exist, the
          dependency is silently discarded.  If a referenced
          subscription is deleted this dependency is removed.";
       reference
         "RFC-7540, section 5.3.1";
     }
   }

   grouping subscription-policy-modifiable {
     description
       "This grouping describes all objects which may be changed
       in a subscription.";
     choice target {
       mandatory true;
       description
         "Identifies the source of information against which a
         subscription is being applied, as well as specifics on the
         subset of information desired from that source.";
       case stream {
         choice stream-filter {
           description
             "An event stream filter can be applied to a subscription.
             That filter will come either referenced from a global list,
             or be provided within the subscription itself.";
           case by-reference {
             description
```

```
              "Apply a filter that has been configured separately.";
            leaf stream-filter-name {
              type stream-filter-ref;
              mandatory true;
              description
                "References an existing event stream filter which is to
                be applied to an event stream for the subscription.";
            }
          }
          case within-subscription {
            description
              "Local definition allows a filter to have the same
              lifecycle as the subscription.";
            uses stream-filter-elements;
          }
        }
      }
    }
    leaf stop-time {
      type yang:date-and-time;
      description
        "Identifies a time after which notification messages for a
        subscription should not be sent.  If 'stop-time' is not
        present, the notification messages will continue until the
        subscription is terminated.  If 'replay-start-time' exists,
        'stop-time' must be for a subsequent time. If
        'replay-start-time' doesn't exist, 'stop-time' when established
        must be for a future time.";
    }
  }

  grouping subscription-policy-dynamic {
    description
      "This grouping describes the only information concerning a
       subscription which can be passed over the RPCs defined in this
       model.";
    uses subscription-policy-modifiable {
      augment target/stream {
        description
          "Adds additional objects which can be modified by RPC.";
        leaf stream {
          type stream-ref {
            require-instance false;
          }
          mandatory true;
          description
            "Indicates the event stream to be considered for
            this subscription.";
```

```
        }
        leaf replay-start-time {
          if-feature "replay";
          type yang:date-and-time;
          config false;
          description
            "Used to trigger the replay feature for a dynamic
            subscription, with event records being selected needing to
            be at or after the start at the time specified.  If
            'replay-start-time' is not present, this is not a replay
            subscription and event record push should start
            immediately. It is never valid to specify start times that
            are later than or equal to the current time.";
        }
      }
    }
    uses update-qos;
  }

  grouping subscription-policy {
    description
      "This grouping describes the full set of policy information
      concerning both dynamic and configured subscriptions, with the
      exclusion of both receivers and networking information specific
      to the publisher such as what interface should be used to
      transmit notification messages.";
    uses subscription-policy-dynamic;
    leaf transport {
      if-feature "configured";
      type transport;
      description
        "For a configured subscription, this leaf specifies the
        transport used to deliver messages destined to all receivers
        of that subscription.";
    }
    leaf encoding {
      when 'not(../transport) or derived-from(../transport,
      "sn:configurable-encoding")';
      type encoding;
      description
        "The type of encoding for notification messages.   For a
        dynamic subscription, if not included as part of an establish-
        subscription RPC, the encoding will be populated with the
        encoding used by that RPC.  For a configured subscription, if
        not explicitly configured the encoding with be the default
        encoding for an underlying transport.";
    }
    leaf purpose {
```

```
        if-feature "configured";
        type string;
        description
          "Open text allowing a configuring entity to embed the
          originator or other specifics of this subscription.";
      }
    }

    /*
     * RPCs
     */

    rpc establish-subscription {
      description
        "This RPC allows a subscriber to create (and possibly negotiate)
         a subscription on its own behalf.  If successful, the
         subscription remains in effect for the duration of the
         subscriber's association with the publisher, or until the
         subscription is terminated. In case an error occurs, or the
         publisher cannot meet the terms of a subscription, an RPC error
         is returned, the subscription is not created.  In that case, the
         RPC reply's 'error-info' MAY include suggested parameter
         settings that would have a higher likelihood of succeeding in a
         subsequent 'establish-subscription' request.";
      input {
        uses subscription-policy-dynamic;
        leaf encoding {
          type encoding;
          description
            "The type of encoding for the subscribed data. If not
            included as part of the RPC, the encoding MUST be set by the
            publisher to be the encoding used by this RPC.";
        }
      }
      output {
        leaf id {
          type subscription-id;
          mandatory true;
          description
            "Identifier used for this subscription.";
        }
        leaf replay-start-time-revision {
          if-feature "replay";
          type yang:date-and-time;
            description
              "If a replay has been requested, this represents the
              earliest time covered by the event buffer for the requested
              event stream.  The value of this object is the
```

```
             'replay-log-aged-time' if it exists.  Otherwise it is the
             'replay-log-creation-time'.  All buffered event records
             after this time will be replayed to a receiver.  This
             object will only be sent if the starting time has been
             revised to be later than the time requested by the
             subscriber.";
        }
      }
    }

    rc:yang-data establish-subscription-stream-error-info {
      container establish-subscription-stream-error-info {
        description
          "If any 'establish-subscription' RPC parameters are
          unsupportable against the event stream, a subscription is not
          created and the RPC error response MUST indicate the reason
          why the subscription failed to be created. This yang-data MAY
          be inserted as structured data within a subscription's RPC
          error response to indicate the failure reason.  This yang-data
          MUST be inserted if hints are to be provided back to the
          subscriber.";
        leaf reason {
          type identityref {
            base establish-subscription-error;
          }
          description
            "Indicates the reason why the subscription has failed to
            be created to a targeted event stream.";
          }
        leaf filter-failure-hint {
          type string;
            description
              "Information describing where and/or why a provided filter
               was unsupportable for a subscription.";
        }
      }
    }

    rpc modify-subscription {
      description
        "This RPC allows a subscriber to modify a dynamic subscription's
         parameters.  If successful, the changed subscription
         parameters remain in effect for the duration of the
         subscription, until the subscription is again modified, or until
         the subscription is terminated.  In case of an error or an
         inability to meet the modified parameters, the subscription is
         not modified and the original subscription parameters remain in
         effect. In that case, the RPC error MAY include 'error-info'
```

```
        suggested parameter hints that would have a high likelihood of
        succeeding in a subsequent 'modify-subscription' request.  A
        successful 'modify-subscription' will return a suspended
        subscription to an 'active' state.";
     input {
       leaf id {
         type subscription-id;
         mandatory true;
         description
           "Identifier to use for this subscription.";
       }
       uses subscription-policy-modifiable;
     }
   }

   rc:yang-data modify-subscription-stream-error-info {
     container modify-subscription-stream-error-info {
       description
         "This yang-data MAY be provided as part of a subscription's RPC
         error response when there is a failure of a
         'modify-subscription' RPC which has been made against an event
         stream.  This yang-data MUST be used if hints are to be
         provided back to the subscriber.";
       leaf reason {
         type identityref {
           base modify-subscription-error;
         }
         description
           "Information in a 'modify-subscription' RPC error response
           which indicates the reason why the subscription to an event
           stream has failed to be modified.";
       }
       leaf filter-failure-hint {
         type string;
           description
             "Information describing where and/or why a provided filter
              was unsupportable for a subscription.";
       }
     }
   }

   rpc delete-subscription {
     description
       "This RPC allows a subscriber to delete a subscription that
        was previously created from by that same subscriber using the
        'establish-subscription' RPC.

        If an error occurs, the server replies with an 'rpc-error' where
```

```
        the 'error-info' field MAY contain an
        'delete-subscription-error-info' structure.";
     input {
       leaf id {
         type subscription-id;
         mandatory true;
         description
           "Identifier of the subscription that is to be deleted.
            Only subscriptions that were created using
            'establish-subscription' from the same origin as this RPC
            can be deleted via this RPC.";
       }
     }
   }

   rpc kill-subscription {
     nacm:default-deny-all;
     description
       "This RPC allows an operator to delete a dynamic subscription
        without restrictions on the originating subscriber or underlying
        transport session.

        If an error occurs, the server replies with an 'rpc-error' where
        the 'error-info' field MAY contain an
        'delete-subscription-error-info' structure.";
     input {
       leaf id {
         type subscription-id;
         mandatory true;
         description
           "Identifier of the subscription that is to be deleted. Only
            subscriptions that were created using
            'establish-subscription' can be deleted via this RPC.";
       }
     }
   }

   rc:yang-data delete-subscription-error-info {
     container delete-subscription-error-info {
       description
         "If a 'delete-subscription' RPC or a 'kill-subscription' RPC
         fails, the subscription is not deleted and the RPC error
         response MUST indicate the reason for this failure. This
         yang-data MAY be inserted as structured data within a
         subscription's RPC error response to indicate the failure
         reason.";
       leaf reason {
         type identityref {
```

```
          base delete-subscription-error;
        }
        mandatory true;
        description
          "Indicates the reason why the subscription has failed to be
          deleted.";
      }
    }
  }

  /*
   * NOTIFICATIONS
   */

  notification replay-completed {
    sn:subscription-state-notification;
    if-feature "replay";
    description
      "This notification is sent to indicate that all of the replay
        notifications have been sent. It must not be sent for any other
       reason.";
    leaf id {
      type subscription-id;
      mandatory true;
      description
        "This references the affected subscription.";
    }
  }

  notification subscription-completed {
    sn:subscription-state-notification;
    if-feature "configured";
    description
      "This notification is sent to indicate that a subscription has
       finished passing event records, as the 'stop-time' has been
       reached.";
    leaf id {
      type subscription-id;
      mandatory true;
      description
        "This references the gracefully completed subscription.";
    }
  }

  notification subscription-modified {
    sn:subscription-state-notification;
    description
      "This notification indicates that a subscription has been
```

```
        modified.  Notification messages sent from this point on will
        conform to the modified terms of the subscription.  For
        completeness, this subscription state change notification
        includes both modified and non-modified aspects of a
        subscription.";
     leaf id {
       type subscription-id;
       mandatory true;
       description
         "This references the affected subscription.";
     }
     uses subscription-policy {
       refine "target/stream/stream-filter/within-subscription" {
         description
           "Filter applied to the subscription.  If the
           'stream-filter-name' is populated, the filter within the
           subscription came from the 'filters' container.  Otherwise it
           is populated in-line as part of the subscription.";
       }
     }
   }

   notification subscription-resumed {
     sn:subscription-state-notification;
     description
       "This notification indicates that a subscription that had
        previously been suspended has resumed. Notifications will once
        again be sent.  In addition, a 'subscription-resumed' indicates
        that no modification of parameters has occurred since the last
        time event records have been sent.";
     leaf id {
       type subscription-id;
       mandatory true;
       description
         "This references the affected subscription.";
     }
   }

   notification subscription-started {
     sn:subscription-state-notification;
     if-feature "configured";
     description
       "This notification indicates that a subscription has started and
         notifications are beginning to be sent. This notification shall
        only be sent to receivers of a subscription; it does not
        constitute a general-purpose notification.";
     leaf id {
       type subscription-id;
```

```
        mandatory true;
        description
          "This references the affected subscription.";
      }
      uses subscription-policy {
        refine "target/stream/replay-start-time" {
           description
             "Indicates the time that a replay using for the streaming of
             buffered event records.  This will be populated with the
             most recent of the following: the event time of the previous
             event record sent to a receiver, the
             'replay-log-creation-time', the 'replay-log-aged-time',
             or the most recent publisher boot time.";
        }
        refine "target/stream/stream-filter/within-subscription" {
          description
            "Filter applied to the subscription.  If the
            'stream-filter-name' is populated, the filter within the
            subscription came from the 'filters' container.  Otherwise it
            is populated in-line as part of the subscription.";
        }
        augment "target/stream" {
          description
            "This augmentation adds additional parameters specific to a
            subscription-started notification.";
          leaf replay-previous-event-time {
            when "../replay-start-time";
            if-feature "replay";
            type yang:date-and-time;
              description
              "If there is at least one event in the replay buffer prior
              to 'replay-start-time', this gives the time of the event
              generated immediately prior to the 'replay-start-time'.

              If a receiver previously received event records for this
              configured subscription, it can compare this time to the
              last event record previously received.  If the two are not
              the same (perhaps due to a reboot), then a dynamic replay
              can be initiated to acquire any missing event records.";
          }
        }
      }
    }

    notification subscription-suspended {
      sn:subscription-state-notification;
      description
        "This notification indicates that a suspension of the
```

```
        subscription by the publisher has occurred.  No further
        notifications will be sent until the subscription resumes.
        This notification shall only be sent to receivers of a
        subscription; it does not constitute a general-purpose
        notification.";
     leaf id {
       type subscription-id;
       mandatory true;
       description
         "This references the affected subscription.";
     }
     leaf reason {
       type identityref {
         base subscription-suspended-reason;
       }
       mandatory true;
       description
         "Identifies the condition which resulted in the suspension.";
     }
   }

   notification subscription-terminated {
     sn:subscription-state-notification;
     description
       "This notification indicates that a subscription has been
        terminated.";
     leaf id {
       type subscription-id;
       mandatory true;
       description
         "This references the affected subscription.";
     }
     leaf reason {
       type identityref {
         base subscription-terminated-reason;
       }
       mandatory true;
       description
         "Identifies the condition which resulted in the termination .";
     }
   }


   /*
    * DATA NODES
    */

   container streams {
```

```
      config false;
      description
        "This container contains information on the built-in event
        streams provided by the publisher.";
      list stream {
        key "name";
        description
          "Identifies the built-in event streams that are supported by
           the publisher.";
        leaf name {
          type string;
          description
            "A handle for a system-provided event stream made up of a
            sequential set of event records, each of which is
            characterized by its own domain and semantics.";
        }
        leaf description {
          type string;
          mandatory true;
          description
            "A description of the event stream, including such
             information as the type of event records that are available
             within this event stream.";
        }
        leaf replay-support {
          if-feature "replay";
          type empty;
          description
            "Indicates that event record replay is available on this
            event stream.";
        }
        leaf replay-log-creation-time {
          when "../replay-support";
          if-feature "replay";
          type yang:date-and-time;
          mandatory true;
          description
            "The timestamp of the creation of the log used to support the
            replay function on this event stream. This time might be
            earlier than the earliest available information contained in
            the log. This object is updated if the log resets for some
            reason.";
        }
        leaf replay-log-aged-time {
          when "../replay-support";
          if-feature "replay";
          type yang:date-and-time;
          description
```

```
           "The timestamp associated with last event record which has
            been aged out of the log.  This timestamp identifies how far
            back into history this replay log extends, if it doesn't
            extend back to the 'replay-log-creation-time'.  This object
            MUST be present if replay is supported and any event records
            have been aged out of the log.";
        }
      }
    }

    container filters {
      description
        "This container contains a list of configurable filters
         that can be applied to subscriptions.  This facilitates
         the reuse of complex filters once defined.";
      list stream-filter {
        key "name";
        description
          "A list of pre-configured filters that can be applied to
          subscriptions.";
        leaf name {
          type string;
          description
            "An name to differentiate between filters.";
        }
        uses stream-filter-elements;
      }
    }

    container subscriptions {
      description
        "Contains the list of currently active subscriptions, i.e.
         subscriptions that are currently in effect, used for
         subscription management and monitoring purposes. This includes
         subscriptions that have been setup via RPC primitives as well as
         subscriptions that have been established via configuration.";
      list subscription {
        key "id";
        description
          "The identity and specific parameters of a subscription.
           Subscriptions within this list can be created using a control
           channel or RPC, or be established through configuration.

           If configuration operations or the 'kill-subscription' RPC are
           used to delete a subscription, a 'subscription-terminated'
           message is sent to any active or suspended receivers.";
        leaf id {
          type subscription-id;
```

```
          description
            "Identifier of a subscription; unique within a publisher";
        }
        uses subscription-policy {
          refine "target/stream/stream" {
            description
              "Indicates the event stream to be considered for this
              subscription.  If an event stream has been removed,
              and no longer can be referenced by an active subscription,
              send a 'subscription-terminated' notification with
              'stream-unavailable' as the reason.  If a configured
              subscription refers to a non-existent event stream, move
              that subscription to the 'invalid' state.";
          }
          refine "transport" {
            description
              "For a configured subscription, this leaf specifies the
              transport used to deliver messages destined to all
              receivers of that subscription.  This object is mandatory
              for subscriptions in the configuration datastore.  This
              object is not mandatory for dynamic subscriptions within
              the operational state datastore.  The object should not
              be present for dynamic subscriptions.";
          }
          augment "target/stream" {
            description
              "Enables objects to added to a configured stream
              subscription";
            leaf configured-replay {
              if-feature "configured";
              if-feature "replay";
              type empty;
              description
                "The presence of this leaf indicates that replay for the
                configured subscription should start at the earliest time
                in the event log, or at the publisher boot time, which
                ever is later.";
            }
          }
        }
        choice notification-message-origin {
          if-feature "configured";
          description
            "Identifies the egress interface on the publisher from which
             notification messages are to be sent.";
          case interface-originated {
            description
              "When notification messages to egress a specific,
```

```
              designated interface on the publisher.";
           leaf source-interface {
             if-feature "interface-designation";
             type if:interface-ref;
             description
               "References the interface for notification messages.";
           }
         }
         case address-originated {
           description
             "When notification messages are to depart from a publisher
              using specific originating address and/or routing context
              information.";
           leaf source-vrf {
             if-feature "supports-vrf";
             type leafref {
               path "/ni:network-instances/ni:network-instance/ni:name";
             }
             description
               "VRF from which notification messages should egress a
               publisher.";
           }
           leaf source-address {
             type inet:ip-address-no-zone;
             description
               "The source address for the notification messages.  If a
               source VRF exists, but this object doesn't, a publisher's
               default address for that VRF must be used.";
           }
         }
       }
       leaf configured-subscription-state {
         if-feature "configured";
         type enumeration {
           enum valid {
             value 1;
             description
               "Subscription is supportable with current parameters.";
           }
           enum invalid {
             value 2;
             description
               "The subscription as a whole is unsupportable with its
               current parameters.";
           }
           enum concluded {
             value 3;
               description
```

```
                 "A subscription is inactive as it has hit a stop time,
                 but not yet been removed from configuration.";
            }
          }
          config false;
          description
            "The presence of this leaf indicates that the subscription
            originated from configuration, not through a control channel
            or RPC.  The value indicates the system established state
            of the subscription.";
        }
        container receivers {
          description
            "Set of receivers in a subscription.";
          list receiver {
            key "name";
            min-elements 1;
            description
              "A host intended as a recipient for the notification
              messages of a subscription.  For configured subscriptions,
              transport specific network parameters (or a leafref to
              those parameters) may augmentated to a specific receiver
              within this list.";
            leaf name {
              type string;
              description
                "Identifies a unique receiver for a subscription.";
            }
            leaf sent-event-records {
              type yang:zero-based-counter64;
              config false;
              description
                "The number of event records sent to the receiver.  The
                count is initialized when a dynamic subscription is
                established, or when a configured receiver
                transitions to the valid state.";
            }
            leaf excluded-event-records {
              type yang:zero-based-counter64;
              config false;
              description
                "The number of event records explicitly removed either
                via an event stream filter or an access control filter so
                that they are not passed to a receiver.  This count is
                set to zero each time 'sent-event-records' is
                initialized.";
            }
            leaf state {
```

```
             type enumeration {
               enum active {
                 value 1;
                 description
                   "Receiver is currently being sent any applicable
                   notification messages for the subscription.";
               }
               enum suspended {
                 value 2;
                 description
                   "Receiver state is 'suspended', so the publisher
                   is currently unable to provide notification messages
                   for the subscription.";
               }
               enum connecting {
                 value 3;
                 if-feature "configured";
                 description
                   "A subscription has been configured, but a
                   'subscription-started' subscription state change
                   notification needs to be successfully received before
                   notification messages are sent.

                   If the 'reset' action is invoked for a receiver of an
                   active configured subscription, the state must be
                   moved to 'connecting'.";
               }
               enum disconnected {
                 value 4;
                 if-feature "configured";
                 description
                   "A subscription has failed in sending a subscription
                   started state change to the receiver.
                   Additional attempts at connection attempts are not
                   currently being made.";
               }
             }
             config false;
             mandatory true;
             description
               "Specifies the state of a subscription from the
               perspective of a particular receiver.  With this info it
               is possible to determine whether a subscriber is
               currently generating notification messages intended for
               that receiver.";
           }
           action reset {
             if-feature "configured";
```

```
            description
              "Allows the reset of this configured subscription
              receiver to the 'connecting' state. This enables the
              connection process to be re-initiated.";
            output {
              leaf time {
                type yang:date-and-time;
                mandatory true;
                description
                  "Time a publisher returned the receiver to a
                  'connecting' state.";
              }
            }
          }
        }
      }
    }
  }
}
<CODE ENDS>
```

5. Considerations

5.1. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications
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 [RFC6020]:

Name: ietf-subscribed-notifications
Namespace: urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications
Prefix: sn
Reference: draft-ietf-netconf-ietf-subscribed-notifications-11.txt
(RFC form)

5.2. Implementation Considerations

To support deployments including both configured and dynamic subscriptions, it is recommended to split the subscription "id" domain into static and dynamic halves. That way it eliminates the possibility of collisions if the configured subscriptions attempt to set a subscription-id which might have already been dynamically allocated. A best practice is to use lower half the "id" object's integer space when that "id" is assigned by an external entity (such as with a configured subscription). This leaves the upper half of subscription integer space available to be dynamically assigned by the publisher.

If a subscription is unable to marshal a series of filtered event records into transmittable notification messages, the receiver should be suspended with the reason "unsupportable-volume".

For configured subscriptions, operations are against the set of receivers using the subscription "id" as a handle for that set. But for streaming updates, subscription state change notifications are local to a receiver. In this specification it is the case that receivers get no information from the publisher about the existence of other receivers. But if a network operator wants to let the receivers correlate results, it is useful to use the subscription "id" across the receivers to allow that correlation.

For configured replay subscriptions, the receiver is protected from duplicated events being pushed after a publisher is rebooted. However it is possible that a receiver might want to acquire event records which failed to be delivered just prior to the reboot. Delivering these event records be accomplished by leveraging the "eventTime" from the last event record received prior to the receipt of a "subscription-started" subscription state change notification. With this "eventTime" and the "replay-start-time" from the "subscription-started" notification, an independent dynamic subscription can be established which retrieves any event records which may have been generated but not sent to the receiver.

5.3. Transport Requirements

This section provides requirements for any subscribed notification transport supporting the solution presented in this document.

The transport selected by the subscriber to reach the publisher MUST be able to support multiple "establish-subscription" requests made within the same transport session.

For both configured and dynamic subscriptions the publisher MUST authenticate a receiver via some transport level mechanism before sending any event records for which they are authorized to see. In addition, the receiver MUST authenticate the publisher at the transport level. The result is mutual authentication between the two.

A secure transport is highly recommended and the publisher MUST ensure that the receiver has sufficient authorization to perform the function they are requesting against the specific subset of content involved.

A specific transport specification built upon this document may or may not choose to require the use of the same logical channel for the RPCs and the event records. However the event records and the subscription state change notifications MUST be sent on the same transport session to ensure the properly ordered delivery.

Additional transport requirements will be dictated by the choice of transport used with a subscription. For an example of such requirements with NETCONF transport, see [I-D.draft-ietf-netconf-netconf-event-notifications].

5.4. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management transports 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) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF operations and content.

One subscription "id" can be used for two or more receivers of the same configured subscription. But due to the possibility of different access control permissions per receiver, it cannot be assumed that each receiver is getting identical updates.

With configured subscriptions, one or more publishers could be used to overwhelm a receiver. Notification messages SHOULD NOT be sent to any receiver which does not support this specification. Receivers that do not want notification messages need only terminate or refuse any transport sessions from the publisher.

When a receiver of a configured subscription gets a new "subscription-started" message for a known subscription where it is already consuming events, the receiver SHOULD retrieve any event records generated since the last event record was received. This can be accomplish by establishing a separate dynamic replay subscription with the same filtering criteria with the publisher, assuming the publisher supports the "replay" feature.

For dynamic subscriptions, implementations need to protect against malicious or buggy subscribers which may send a large number "establish-subscription" requests, thereby using up system resources. To cover this possibility operators SHOULD monitor for such cases and, if discovered, take remedial action to limit the resources used, such as suspending or terminating a subset of the subscriptions or, if the underlying transport is session based, terminate the underlying transport session.

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 where there is a specific sensitivity/vulnerability:

Container: "/filters"

- o "stream-subtree-filter": updating a filter could increase the computational complexity of all referencing subscriptions.

- o "stream-xpath-filter": updating a filter could increase the computational complexity of all referencing subscriptions.

Container: "/subscriptions"

The following considerations are only relevant for configuration operations made upon configured subscriptions:

- o "configured-replay": can be used to send a large number of event records to a receiver.

- o "dependency": can be used to force important traffic to be queued behind less important updates.

- o "dscp": if unvalidated, can result in the sending of traffic with a higher priority marking than warranted.

- o "id": can overwrite an existing subscription, perhaps one configured by another entity.

- o "name": adding a new key entry can be used to attempt to send traffic to an unwilling receiver.

- o "replay-start-time": can be used to push very large logs, wasting resources.

- o "source-address": the configured address might not be able to reach a desired receiver.

- o "source-interface": the configured interface might not be able to reach a desired receiver.

- o "source-vrf": can place a subscription into a virtual network where receivers are not entitled to view the subscribed content.

- o "stop-time": could be used to terminate content at an inopportune time.

- o "stream": could set a subscription to an event stream containing no content permitted for the targeted receivers.

- o "stream-filter-name": could be set to a filter which is irrelevant to the event stream.

- o "stream-subtree-filter": a complex filter can increase the computational resources for this subscription.

- o "stream-xpath-filter": a complex filter can increase the computational resources for this subscription.

- o "weighting": placing a large weight can overwhelm the dequeuing of other subscriptions.

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:

Container: "/streams"

- o "name": if access control is not properly configured, can expose system internals to those who should have no access to this information.

- o "replay-support": if access control is not properly configured, can expose logs to those who should have no access.

Container: "/subscriptions"

o "excluded-event-records": leaf can provide information about filtered event records. A network operator should have permissions to know about such filtering.

o "subscription": different operational teams might have a desire to set varying subsets of subscriptions. Access control should be designed to permit read access to just the allowed set.

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:

RPC: all

o If a malicious or buggy subscriber sends an unexpectedly large number of RPCs, the result might be an excessive use of system resources on the publisher just to determine that these subscriptions should be declined. In such a situation, subscription interactions MAY be terminated by terminating the transport session.

RPC: "delete-subscription"

o No special considerations.

RPC: "establish-subscription"

o Subscriptions could overload a publisher's resources. For this reason, publishers MUST ensure that they have sufficient resources to fulfill this request or otherwise reject the request.

RPC: "kill-subscription"

o The "kill-subscription" RPC MUST be secured so that only connections with administrative rights are able to invoke this RPC.

RPC: "modify-subscription"

o Subscriptions could overload a publisher's resources. For this reason, publishers MUST ensure that they have sufficient resources to fulfill this request or otherwise reject the request.

6. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Andy Bierman, Tim Jenkins, Martin Bjorklund, Kent Watsen, Balazs Lengyel, Robert Wilton, Sharon Chisholm, Hector Trevino, Susan Hares, Michael Scharf, and Guangying Zheng.

7. References

7.1. Normative References

[I-D.draft-ietf-rtgwg-ni-model]
Berger, L., Hopps, C., and A. Lindem, "YANG Network Instances", draft-ietf-rtgwg-ni-model-12 (work in progress), March 2018.

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

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

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

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

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

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

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[XPATH] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", November 1999, <http://www.w3.org/TR/1999/REC-xpath-19991116>.

7.2. Informative References

[I-D.draft-ietf-netconf-netconf-event-notifications]
Clemm, Alexander., Voit, Eric., Gonzalez Prieto, Alberto., Nilsen-Nygaard, E., and A. Tripathy, "NETCONF support for event notifications", May 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-netconf-event-notifications/>.

[I-D.draft-ietf-netconf-restconf-notif]
Voit, Eric., Clemm, Alexander., Tripathy, A., Nilsen-Nygaard, E., and Alberto. Gonzalez Prieto, "Restconf and HTTP transport for event notifications", May 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-restconf-notif/>.

[I-D.ietf-netconf-yang-push]
Clemm, Alexander., Voit, Eric., Gonzalez Prieto, Alberto., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", May 2018, <https://datatracker.ietf.org/doc/draft-ietf-netconf-yang-push/>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <https://www.rfc-editor.org/info/rfc7923>.

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Example Configured Transport Augmentation

This appendix provides a non-normative example of how the YANG model defined in Section 4 may be enhanced to incorporate the configuration parameters needed to support the transport connectivity process. In this example, connectivity via an imaginary transport type of "foo" is explored. For more on the overall need, see Section 2.5.7.

The YANG model defined in this section contains two main elements. First is a transport identity "foo". This transport identity allows a configuration agent to define "foo" as the selected type of transport for a subscription. Second is a YANG case augmentation "foo" which is made to the "/subscriptions/subscription/receivers/receiver" node of Section 4. Within this augmentation are the transport configuration parameters "address" and "port" which are necessary to make the connect to the receiver.

```
module example-foo-subscribed-notifications {
  yang-version 1.1;
  namespace
    "urn:example:foo-subscribed-notifications";

  prefix fsn;

  import ietf-subscribed-notifications {
    prefix sn;
  }
  import ietf-inet-types {
    prefix inet;
  }

  description
    "Defines 'foo' as a supported type of configured transport for
    subscribed event notifications.";

  identity foo {
    base sn:transport;
    description
      "Transport type 'foo' is available for use as a configured
       subscription transport protocol for subscribed notifications.";
  }

  augment
    "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
    when 'derived-from(../../../transport, "fsn:foo")';
    description
      "This augmentation makes 'foo' specific transport parameters
      available  for a receiver.";
    leaf address {
      type inet:host;
      mandatory true;
      description
        "Specifies the address to use for messages destined to a
        receiver.";
    }
    leaf port {
```

```
        type inet:port-number;
        mandatory true;
        description
          "Specifies the port number to use for messages destined to a
          receiver.";
      }
    }
  }
```

Figure 21: Example Transport Augmentation for the fictitious protocol foo

This example YANG model for transport "foo" will not be seen in a real world deployment. For a real world deployment supporting an actual transport technology, a similar YANG model must be defined.

Appendix B. Changes between revisions

(To be removed by RFC editor prior to publication)

v17 - v18

- o Transport optional in YANG model.
- o Modify subscription must come from the originator of the subscription. (Text got dropped somewhere previously.)
- o Title change.

v16 - v17

- o YANG renaming: Subscription identifier renamed to id. Counters renamed. Filters id made into name.
- o Text tweaks.

v15 - v16

- o Mandatory empty case "transport" removed.
- o Appendix case turned from "netconf" to "foo".

v14 - v15

- o Text tweaks.
- o Mandatory empty case "transport" added for transport parameters. This includes a new section and an appendix explaining it.

v13 - v14

o Removed the 'address' leaf.

o Replay is now of type 'empty' for configured.

v12 - v13

o Tweaks from Kent's comments

o Referenced in YANG model updated per Tom Petch's comments

o Added leaf replay-previous-event-time

o Renamed the event counters, downshifted the subscription states

v11 - v12

o Tweaks from Kent's, Tim's, and Martin's comments

o Clarified dscp text, and made its own feature

o YANG model tweaks alphabetizing, features.

v10 - v11

o access control filtering of events in streams included to match RFC5277 behavior

o security considerations updated based on YANG template.

o dependency QoS made non-normative on HTTP2 QoS

o tree diagrams referenced for each figure using them

o reference numbers placed into state machine figures

o broke configured replay into its own section

o many tweaks updates based on LC and YANG doctor reviews

o trees and YANG model reconciled were deltas existed

o new feature for interface originated.

o dscp removed from the qos feature

o YANG model updated in a way which collapses groups only used once so that they are part of the 'subscriptions' container.

o alternative encodings only allowed for transports which support them.

v09 - v10

o Typos and tweaks

v08 - v09

o NMDA model supported. Non NMDA version at https://github.com/netconf-wg/rfc5277bis/

o Error mechanism revamped to match to embedded implementations.

o Explicitly identified error codes relevant to each RPC/Notification

v07 - v08

o Split YANG trees to separate document subsections.

o Clarified configured state machine based on Balazs comments, and moved it into the configured subscription subsections.

o Normative reference to Network Instance model for VRF

o One transport for all receivers of configured subscriptions.

o QoS section moved in from yang-push

v06 - v07

o Clarification on state machine for configured subscriptions.

v05 - v06

o Made changes proposed by Martin, Kent, and others on the list. Most significant of these are stream returned to string (with the SYSLOG identity removed), intro section on 5277 relationship, an identity set moved to an enumeration, clean up of definitions/terminology, state machine proposed for configured subscriptions with a clean-up of subscription state options.

o JSON and XML become features. Also Xpath and subtree filtering become features

- o Terminology updates with event records, and refinement of filters to just event stream filters.

- o Encoding refined in establish-subscription so it takes the RPC's encoding as the default.

- o Namespaces in examples fixed.

v04 - v05

- o Returned to the explicit filter subtyping of v00

- o stream object changed to 'name' from 'stream'

- o Cleaned up examples

- o Clarified that JSON support needs notification-messages draft.

v03 - v04

- o Moved back to the use of RFC5277 one-way notifications and encodings.

v03 - v04

- o Replay updated

v02 - v03

- o RPCs and Notification support is identified by the Notification 2.0 capability.

- o Updates to filtering identities and text

- o New error type for unsupportable volume of updates

- o Text tweaks.

v01 - v02

- o Subscription status moved under receiver.

v00 - v01

- o Security considerations updated

- o Intro rewrite, as well as scattered text changes

- o Added Appendix A, to help match this to related drafts in progress
- o Updated filtering definitions, and filter types in yang file, and moved to identities for filter types
- o Added Syslog as an event stream
- o HTTP2 moved in from YANG-Push as a transport option
- o Replay made an optional feature for events. Won't apply to datastores
- o Enabled notification timestamp to have different formats.
- o Two error codes added.

v01 5277bis - v00 subscribed notifications

- o Kill subscription RPC added.
- o Renamed from 5277bis to Subscribed Notifications.
- o Changed the notification capabilities version from 1.1 to 2.0.
- o Extracted create-subscription and other elements of RFC5277.
- o Error conditions added, and made specific in return codes.
- o Simplified yang model structure for removal of 'basic' grouping.
- o Added a grouping for items which cannot be statically configured.
- o Operational counters per receiver.
- o Subscription-id and filter-id renamed to identifier
- o Section for replay added. Replay now cannot be configured.
- o Control plane notification renamed to subscription state change notification
- o Source address: Source-vrf changed to string, default address option added
- o In yang model: 'info' changed to 'policy'
- o Scattered text clarifications

v00 - v01 of 5277bis

- o YANG Model changes. New groupings for subscription info to allow restriction of what is changeable via RPC. Removed notifications for adding and removing receivers of configured subscriptions.

- o Expanded/renamed definitions from event server to publisher, and client to subscriber as applicable. Updated the definitions to include and expand on RFC 5277.

- o Removal of redundancy with other drafts

- o Many other clean-ups of wording and terminology

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Alberto Gonzalez Prieto
Microsoft

Email: alberto.gonzalez@microsoft.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Ambika Prasad Tripathy
Cisco Systems

Email: ambtripa@cisco.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

K. Watsen
Juniper Networks
G. Wu
Cisco Systems
L. Xia
Huawei

October 22, 2018

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-08

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:

o "2018-10-22" --> the publication date of this draft

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

o Appendix A. Change Log

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Terminology 4
3. The TLS Client Model 4
  3.1. Tree Diagram 4
  3.2. Example Usage 4
  3.3. YANG Module 6
4. The TLS Server Model 9
  4.1. Tree Diagram 9

4.2. Example Usage 10
4.3. YANG Module 12
5. The TLS Common Model 15
5.1. Tree Diagram 24
5.2. Example Usage 24
5.3. YANG Module 24
6. Security Considerations 33
7. IANA Considerations 34
7.1. The IETF XML Registry 34
7.2. The YANG Module Names Registry 34
8. References 35
8.1. Normative References 35
8.2. Informative References 36
Appendix A. Change Log 38
A.1. 00 to 01 38
A.2. 01 to 02 38
A.3. 02 to 03 38
A.4. 03 to 04 38
A.5. 04 to 05 39
A.6. 05 to 06 39
A.7. 06 to 07 39
A.8. 07 to 08 39
Acknowledgements 39
Authors' Addresses 39

1. Introduction

This document defines three YANG 1.1 [RFC7950] modules: the first defines a grouping for a generic TLS client, the second defines a grouping for a generic TLS server, and the third defines identities and groupings common to both the client and the server (TLS is defined in [RFC5246]). It is intended that these groupings will be used by applications using the TLS protocol. For instance, these groupings could be used to help define the data model for an HTTPS [RFC2818] server or a NETCONF over TLS [RFC7589] based server.

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

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

2. Terminology

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

3. The TLS Client Model

3.1. Tree Diagram

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

```
module: ietf-tls-client

  grouping server-auth-grouping
    +-- server-auth
       +-- pinned-ca-certs?       ta:pinned-certificates-ref
       |       {ta:x509-certificates}?
       +-- pinned-server-certs?   ta:pinned-certificates-ref
               {ta:x509-certificates}?
  grouping tls-client-grouping
    +---u client-identity-grouping
    +---u server-auth-grouping
    +---u hello-params-grouping
  grouping client-identity-grouping
    +-- client-identity
       +-- (auth-type)?
          +--:(certificate)
             +-- certificate
                +---u client-identity-grouping
  grouping hello-params-grouping
    +-- hello-params {tls-client-hello-params-config}?
       +---u hello-params-grouping
```

3.2. Example Usage

This section presents two examples showing the tls-client-grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 3 of

[I-D.ietf-netconf-trust-anchors] and Section 3.2 of [I-D.ietf-netconf-keystore].

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

```
[Note: '\' line wrapping for formatting only]

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

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <certificate>
      <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-t\
ypes">ct:rsa2048</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
      <cert>base64encodedvalue==</cert>
    </certificate>
  </client-identity>

  <!-- which certificates will this client trust -->
  <server-auth>
    <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinned-ca-c\
erts>
    <pinned-server-certs>explicitly-trusted-server-certs</pinned-ser\
ver-certs>
  </server-auth>

</tls-client>
```

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

[Note: '\' line wrapping for formatting only]

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

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <certificate>
      <reference>ex-rsa-cert</reference>
    </certificate>
  </client-identity>

  <!-- which certificates will this client trust -->
  <server-auth>
    <pinned-ca-certs>explicitly-trusted-server-ca-certs</pinned-ca-c\
erts>
    <pinned-server-certs>explicitly-trusted-server-certs</pinned-ser\
ver-certs>
  </server-auth>

</tls-client>
```

3.3. YANG Module

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

```
<CODE BEGINS> file "ietf-tls-client@2018-10-22.yang"
module ietf-tls-client {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix "tlsc";

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  import ietf-keystore {
    prefix ks;
```

```
       reference
         "RFC ZZZZ: YANG Data Model for a 'Keystore' Mechanism";
     }

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

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

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>

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


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

       Copyright (c) 2018 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       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.";
```

```
    }

    // groupings

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

    grouping client-identity-grouping {
      description
        "A reusable grouping for configuring a TLS client identity.";
      container client-identity {
        description
          "The credentials used by the client to authenticate to
           the TLS server.";

        choice auth-type {
          description
            "The authentication type.";
          container certificate {
            uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
            description
              "A locally-defined or referenced certificate
               to be used for client authentication.";
            reference
              "RFC ZZZZ: YANG Data Model for a 'Keystore' Mechanism";
          }
        }
      } // end client-identity
    } // end client-identity-grouping

    grouping server-auth-grouping {
      description
        "A reusable grouping for configuring TLS server
         authentication.";
      container server-auth {
        must 'pinned-ca-certs or pinned-server-certs';
        description
          "Trusted server identities.";
        leaf pinned-ca-certs {
          if-feature "ta:x509-certificates";
          type ta:pinned-certificates-ref;
```

```
          description
            "A reference to a list of certificate authority (CA)
             certificates used by the TLS client to authenticate
             TLS server certificates.  A server certificate is
             authenticated if it has a valid chain of trust to
             a configured pinned CA certificate.";
        }
        leaf pinned-server-certs {
          if-feature "ta:x509-certificates";
          type ta:pinned-certificates-ref;
          description
            "A reference to a list of server certificates used by
             the TLS client to authenticate TLS server certificates.
             A server certificate is authenticated if it is an
             exact match to a configured pinned server certificate.";
        }
      }
    } // end server-auth-grouping

    grouping hello-params-grouping {
      description
        "A reusable grouping for configuring a TLS transport
         parameters.";
      container hello-params {
        if-feature tls-client-hello-params-config;
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      }
    } // end transport-params-grouping

  }
  <CODE ENDS>
```

4. The TLS Server Model

4.1. Tree Diagram

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

```
module: ietf-tls-server

  grouping hello-params-grouping
    +-- hello-params {tls-server-hello-params-config}?
       +---u hello-params-grouping
  grouping server-identity-grouping
    +-- server-identity
       +---u server-identity-grouping
  grouping tls-server-grouping
    +---u server-identity-grouping
    +---u client-auth-grouping
    +---u hello-params-grouping
  grouping client-auth-grouping
    +-- client-auth
       +-- pinned-ca-certs?       ta:pinned-certificates-ref
       |       {ta:x509-certificates}?
       +-- pinned-client-certs?   ta:pinned-certificates-ref
               {ta:x509-certificates}?
```

4.2. Example Usage

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

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

[Note: '\' line wrapping for formatting only]

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

  <!-- how this server will authenticate itself to the client -->
  <server-identity>
    <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-typ\
es">ct:rsa2048</algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <cert>base64encodedvalue==</cert>
  </server-identity>

  <!-- which certificates will this server trust -->
  <client-auth>
    <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
erts>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
ent-certs>
  </client-auth>

</tls-server>
```

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

[Note: '\' line wrapping for formatting only]

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

  <!-- how this server will authenticate itself to the client -->
  <server-identity>
    <reference>ex-rsa-cert</reference>
  </server-identity>

  <!-- which certificates will this server trust -->
  <client-auth>
    <pinned-ca-certs>explicitly-trusted-client-ca-certs</pinned-ca-c\
erts>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
ent-certs>
  </client-auth>

</tls-server>
```

4.3. YANG Module

This YANG module has a normative references to [RFC5246], [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore].

```
<CODE BEGINS> file "ietf-tls-server@2018-10-22.yang"
module ietf-tls-server {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix "tlss";

  import ietf-tls-common {
    prefix tlscmn;
    revision-date 2018-10-22; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
  }

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

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

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

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

    Author:   Kent Watsen
              <mailto:kwatsen@juniper.net>

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


  description
   "This module defines a reusable grouping for a TLS server that
    can be used as a basis for specific TLS server instances.
```

```
       Copyright (c) 2018 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       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.";
     }


     // groupings

     grouping tls-server-grouping {
       description
         "A reusable grouping for configuring a TLS server without
          any consideration for how underlying TCP sessions are
          established.";
       uses server-identity-grouping;
       uses client-auth-grouping;
       uses hello-params-grouping;
     }

     grouping server-identity-grouping {
       description
         "A reusable grouping for configuring a TLS server identity.";
       container server-identity {
         description
           "A locally-defined or referenced end-entity certificate,
            including any configured intermediate certificates, the
```

```
            TLS server will present when establishing a TLS connection
            in its Certificate message, as defined in Section 7.4.2
            in RFC 5246.";
         reference
           "RFC 5246:
              The Transport Layer Security (TLS) Protocol Version 1.2
            RFC ZZZZ:
              YANG Data Model for a 'Keystore' Mechanism";
         uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
       }
     } // end server-identity-grouping

     grouping client-auth-grouping {
       description
         "A reusable grouping for configuring a TLS client
          authentication.";
       container client-auth {
         description
           "A reference to a list of pinned certificate authority (CA)
            certificates and a reference to a list of pinned client
            certificates.";
         leaf pinned-ca-certs {
           if-feature "ta:x509-certificates";
           type ta:pinned-certificates-ref;
           description
             "A reference to a list of certificate authority (CA)
              certificates used by the TLS server to authenticate
              TLS client certificates.  A client certificate is
              authenticated if it has a valid chain of trust to
              a configured pinned CA certificate.";
           reference
             "RFC YYYY: YANG Data Model for Global Trust Anchors";
         }
         leaf pinned-client-certs {
           if-feature "ta:x509-certificates";
           type ta:pinned-certificates-ref;
           description
             "A reference to a list of client certificates used by
              the TLS server to authenticate TLS client certificates.
              A clients certificate is authenticated if it is an
              exact match to a configured pinned client certificate.";
           reference
             "RFC YYYY: YANG Data Model for Global Trust Anchors";
         }
       }
     } // end client-auth-grouping

     grouping hello-params-grouping {
```

```
        description
          "A reusable grouping for configuring a TLS transport
           parameters.";
        container hello-params {
          if-feature tls-server-hello-params-config;
          uses tlscmn:hello-params-grouping;
          description
            "Configurable parameters for the TLS hello message.";
        }

      } // end tls-server-grouping

    }
    <CODE ENDS>
```

5. The TLS Common Model

The TLS common model presented in this section contains identities and groupings common to both TLS clients and TLS servers. The hello-params-grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the 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;

- TLS 1.3 defines its supported algorithms differently. Firstly, it defines three categories of registries for cryptographic algorithms: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Secondly, all three of these categories are useful, since they represent different parts of all the supported algorithms respectively. Thus, all of these registries categories are considered here. In this draft, the TLS common model chooses only those TLS1.3 algorithms specified in B.4, 4.2.3, 4.2.7 of [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the hello-params-grouping should include three parameters for configuring its permitted TLS algorithms, which are: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups. Note that TLS1.2 only uses TLS Cipher Suites.

[I-D.ietf-netconf-crypto-types] defines six categories of cryptographic algorithms (hash-algorithm, symmetric-key-encryption-algorithm, mac-algorithm, asymmetric-key-encryption-algorithm, signature-algorithm, key-negotiation-algorithm) and lists several widely accepted algorithms for each of them. The TLS client and server models use one or more of these algorithms. The following tables are provided, in part to define the subset of algorithms defined in the crypto-types model used by TLS, and in part to ensure compatibility of configured TLS cryptographic parameters for configuring its permitted TLS algorithms:

ciper-suites in hello-params-grouping	HASH
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	sha-256
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	sha-384
TLS_DHE_RSA_WITH_AES_128_CCM	sha-256
TLS_DHE_RSA_WITH_AES_256_CCM	sha-256
TLS_DHE_PSK_WITH_AES_128_CCM	sha-256
TLS_DHE_PSK_WITH_AES_256_CCM	sha-256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	sha-256
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	sha-256
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	sha-384
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	sha-256

Table 1-1 TLS 1.2 Compatibility Matrix Part 1: ciper-suites mapping to hash-algorithm

ciper-suites in hello-params-grouping	symmetric
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_DHE_RSA_WITH_AES_128_CCM	enc-aes-128-ccm
TLS_DHE_RSA_WITH_AES_256_CCM	enc-aes-256-ccm
TLS_DHE_PSK_WITH_AES_128_CCM	enc-aes-128-ccm
TLS_DHE_PSK_WITH_AES_256_CCM	enc-aes-256-ccm
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	enc-aes-256-gcm
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	enc-aes-128-ccm

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

ciper-suites in hello-params-grouping	MAC
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_DHE_RSA_WITH_AES_128_CCM	mac-aes-128-ccm
TLS_DHE_RSA_WITH_AES_256_CCM	mac-aes-256-ccm
TLS_DHE_PSK_WITH_AES_128_CCM	mac-aes-128-ccm
TLS_DHE_PSK_WITH_AES_256_CCM	mac-aes-256-ccm
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	mac-aes-256-gcm
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	mac-aes-128-ccm

Table 1-3 TLS 1.2 Compatibility Matrix Part 3: ciper-suites mapping to MAC-algorithm

ciper-suites in hello-params-grouping	signature
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	rsa-pkcs1-sha256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	rsa-pkcs1-sha384
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	N/A
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	N/A
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ecdsa-secp256r1-sha256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ecdsa-secp384r1-sha384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	rsa-pkcs1-sha384
TLS_DHE_RSA_WITH_AES_128_CCM	rsa-pkcs1-sha256
TLS_DHE_RSA_WITH_AES_256_CCM	rsa-pkcs1-sha256
TLS_DHE_PSK_WITH_AES_128_CCM	N/A
TLS_DHE_PSK_WITH_AES_256_CCM	N/A
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	ecdsa-secp256r1-sha256
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	rsa-pkcs1-sha256
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	N/A
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	N/A
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	N/A
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	N/A
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	N/A

Table 1-4 TLS 1.2 Compatibility Matrix Part 4: ciper-suites mapping to signature-algorithm

ciper-suites in hello-params-grouping	key-negotiation
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	dhe-ffdhe2048, ...
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	psk-dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	psk-dhe-ffdhe2048, ...
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ecdhe-secp256r1, ...
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	ecdhe-secp256r1, ...
TLS_DHE_RSA_WITH_AES_128_CCM	dhe-ffdhe2048, ...
TLS_DHE_RSA_WITH_AES_256_CCM	dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_128_CCM	psk-dhe-ffdhe2048, ...
TLS_DHE_PSK_WITH_AES_256_CCM	psk-dhe-ffdhe2048, ...
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256	ecdhe-secp256r1, ...
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256	ecdhe-secp256r1, ...
TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256	dhe-ffdhe2048, ...
TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256	psk-ecdhe-secp256r1,...
TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256	psk-dhe-ffdhe2048, ...
TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256	psk-ecdhe-secp256r1,...
TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384	psk-ecdhe-secp256r1,...
TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256	psk-ecdhe-secp256r1,...

Table 1-5 TLS 1.2 Compatibility Matrix Part 5: ciper-suites mapping to key-negotiation-algorithm

ciper-suites in hello -params-grouping	HASH
TLS_AES_128_GCM_SHA256	sha-256
TLS_AES_256_GCM_SHA384	sha-384
TLS_CHACHA20_POLY1305_SHA256	sha-256
TLS_AES_128_CCM_SHA256	sha-256

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

ciper-suites in hello -params-grouping	symmetric
TLS_AES_128_GCM_SHA256	enc-aes-128-gcm
TLS_AES_256_GCM_SHA384	enc-aes-128-gcm
TLS_CHACHA20_POLY1305_SHA256	enc-chacha20-poly1305
TLS_AES_128_CCM_SHA256	enc-aes-128-ccm

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

ciper-suites in hello -params-grouping	symmetric
TLS_AES_128_GCM_SHA256	mac-aes-128-gcm
TLS_AES_256_GCM_SHA384	mac-aes-128-gcm
TLS_CHACHA20_POLY1305_SHA256	mac-chacha20-poly1305
TLS_AES_128_CCM_SHA256	mac-aes-128-ccm

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

signatureScheme in hello -params-grouping	signature
rsa-pkcs1-sha256	rsa-pkcs1-sha256
rsa-pkcs1-sha384	rsa-pkcs1-sha384
rsa-pkcs1-sha512	rsa-pkcs1-sha512
rsa-pss-rsae-sha256	rsa-pss-rsae-sha256
rsa-pss-rsae-sha384	rsa-pss-rsae-sha384
rsa-pss-rsae-sha512	rsa-pss-rsae-sha512
rsa-pss-pss-sha256	rsa-pss-pss-sha256
rsa-pss-pss-sha384	rsa-pss-pss-sha384
rsa-pss-pss-sha512	rsa-pss-pss-sha512
ecdsa-secp256r1-sha256	ecdsa-secp256r1-sha256
ecdsa-secp384r1-sha384	ecdsa-secp384r1-sha384
ecdsa-secp521r1-sha512	ecdsa-secp521r1-sha512
ed25519	ed25519
ed448	ed448

Table 2-4 TLS 1.3 Compatibility Matrix Part 4: SignatureScheme mapping to signature-algorithm

supported Groups in hello -params-grouping	key-negotiation
dhe-ffdhe2048 dhe-ffdhe3072 dhe-ffdhe4096 dhe-ffdhe6144 dhe-ffdhe8192 psk-dhe-ffdhe2048 psk-dhe-ffdhe3072 psk-dhe-ffdhe4096 psk-dhe-ffdhe6144 psk-dhe-ffdhe8192 ecdhe-secp256r1 ecdhe-secp384r1 ecdhe-secp521r1 ecdhe-x25519 ecdhe-x448 psk-ecdhe-secp256r1 psk-ecdhe-secp384r1 psk-ecdhe-secp521r1 psk-ecdhe-x25519 psk-ecdhe-x448	dhe-ffdhe2048 dhe-ffdhe3072 dhe-ffdhe4096 dhe-ffdhe6144 dhe-ffdhe8192 psk-dhe-ffdhe2048 psk-dhe-ffdhe3072 psk-dhe-ffdhe4096 psk-dhe-ffdhe6144 psk-dhe-ffdhe8192 ecdhe-secp256r1 ecdhe-secp384r1 ecdhe-secp521r1 ecdhe-x25519 ecdhe-x448 psk-ecdhe-secp256r1 psk-ecdhe-secp384r1 psk-ecdhe-secp521r1 psk-ecdhe-x25519 psk-ecdhe-x448

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

Note that in Table 1-5:

- 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"
   xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
  <tls-versions>
    <tls-version>tlscmn:tls-1.1</tls-version>
    <tls-version>tlscmn:tls-1.2</tls-version>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>tlscmn:dhe-rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>tlscmn:rsa-with-aes-128-cbc-sha</cipher-suite>
    <cipher-suite>tlscmn:rsa-with-3des-ede-cbc-sha</cipher-suite>
  </cipher-suites>
</hello-params>
```

5.3. YANG Module

This YANG module has a normative references to [RFC2246], [RFC4346], [RFC5246], [RFC5288], [RFC5289], and [RFC8422].

This YANG module has a informative references to [RFC2246], [RFC4346], and [RFC5246].

```
<CODE BEGINS> file "ietf-tls-common@2018-10-22.yang"
module ietf-tls-common {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
  prefix "tlscmn";

  organization
```

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

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

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>

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


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

       Copyright (c) 2018 IETF Trust and the persons identified as
       authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
     }

     // features

     feature tls-1_0 {
       description
         "TLS Protocol Version 1.0 is supported.";
       reference
         "RFC 2246: The TLS Protocol Version 1.0";
     }

     feature tls-1_1 {
       description
```

```
         "TLS Protocol Version 1.1 is supported.";
       reference
         "RFC 4346: The Transport Layer Security (TLS) Protocol
                    Version 1.1";
     }

     feature tls-1_2 {
       description
         "TLS Protocol Version 1.2 is supported.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     feature tls-ecc {
       description
         "Elliptic Curve Cryptography (ECC) is supported for TLS.";
       reference
         "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                    for Transport Layer Security (TLS)";
     }

     feature tls-dhe {
       description
         "Ephemeral Diffie-Hellman key exchange is supported for TLS.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     feature tls-3des {
       description
         "The Triple-DES block cipher is supported for TLS.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     feature tls-gcm {
       description
         "The Galois/Counter Mode authenticated encryption mode is
          supported for TLS.";
       reference
         "RFC 5288: AES Galois Counter Mode (GCM) Cipher Suites for
                    TLS";
     }

     feature tls-sha2 {
```

```
       description
         "The SHA2 family of cryptographic hash functions is supported
          for TLS.";
       reference
         "FIPS PUB 180-4: Secure Hash Standard (SHS)";
     }

     // identities

     identity tls-version-base {
       description
         "Base identity used to identify TLS protocol versions.";
     }

     identity tls-1.0 {
       base tls-version-base;
       if-feature tls-1_0;
       description
         "TLS Protocol Version 1.0.";
       reference
         "RFC 2246: The TLS Protocol Version 1.0";
     }

     identity tls-1.1 {
       base tls-version-base;
       if-feature tls-1_1;
       description
         "TLS Protocol Version 1.1.";
       reference
         "RFC 4346: The Transport Layer Security (TLS) Protocol
                    Version 1.1";
     }

     identity tls-1.2 {
       base tls-version-base;
       if-feature tls-1_2;
       description
         "TLS Protocol Version 1.2.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity cipher-suite-base {
       description
         "Base identity used to identify TLS cipher suites.";
     }
```

```
     identity rsa-with-aes-128-cbc-sha {
       base cipher-suite-base;
       description
         "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity rsa-with-aes-256-cbc-sha {
       base cipher-suite-base;
       description
         "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity rsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature tls-sha2;
       description
         "Cipher suite TLS_RSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity rsa-with-aes-256-cbc-sha256 {
       base cipher-suite-base;
       if-feature tls-sha2;
       description
         "Cipher suite TLS_RSA_WITH_AES_256_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity dhe-rsa-with-aes-128-cbc-sha {
       base cipher-suite-base;
       if-feature tls-dhe;
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }
```

```
     identity dhe-rsa-with-aes-256-cbc-sha {
       base cipher-suite-base;
       if-feature tls-dhe;
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity dhe-rsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-dhe and tls-sha2";
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity dhe-rsa-with-aes-256-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-dhe and tls-sha2";
       description
         "Cipher suite TLS_DHE_RSA_WITH_AES_256_CBC_SHA256.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity ecdhe-ecdsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-ecdsa-with-aes-256-cbc-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
```

```
     }

     identity ecdhe-rsa-with-aes-128-cbc-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-rsa-with-aes-256-cbc-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-ecdsa-with-aes-128-gcm-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-ecdsa-with-aes-256-gcm-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-rsa-with-aes-128-gcm-sha256 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.";
       reference
```

```
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity ecdhe-rsa-with-aes-256-gcm-sha384 {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-gcm and tls-sha2";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.";
       reference
         "RFC 5289: TLS Elliptic Curve Cipher Suites with
                    SHA-256/384 and AES Galois Counter Mode (GCM)";
     }

     identity rsa-with-3des-ede-cbc-sha {
       base cipher-suite-base;
       if-feature tls-3des;
       description
         "Cipher suite TLS_RSA_WITH_3DES_EDE_CBC_SHA.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";
     }

     identity ecdhe-rsa-with-3des-ede-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-ecc and tls-3des";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA.";
       reference
         "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                    for Transport Layer Security (TLS)";
     }

     identity ecdhe-rsa-with-aes-128-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-ecc";
       description
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA.";
       reference
         "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                    for Transport Layer Security (TLS)";
     }

     identity ecdhe-rsa-with-aes-256-cbc-sha {
       base cipher-suite-base;
       if-feature "tls-ecc";
       description
```

```
         "Cipher suite TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA.";
       reference
         "RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites
                    for Transport Layer Security (TLS)";
     }

     // groupings

     grouping hello-params-grouping {
       description
         "A reusable grouping for TLS hello message parameters.";
       reference
         "RFC 5246: The Transport Layer Security (TLS) Protocol
                    Version 1.2";

       container tls-versions {
         description
           "Parameters regarding TLS versions.";
         leaf-list tls-version {
           type identityref {
             base tls-version-base;
           }
           description
             "Acceptable TLS protocol versions.

              If this leaf-list is not configured (has zero elements)
              the acceptable TLS protocol versions are implementation-
              defined.";
         }
       }
       container cipher-suites {
         description
           "Parameters regarding cipher suites.";
         leaf-list cipher-suite {
           type identityref {
             base cipher-suite-base;
           }
           ordered-by user;
           description
             "Acceptable cipher suites in order of descending
              preference.  The configured host key algorithms should
              be compatible with the algorithm used by the configured
              private key.  Please see Section 5 of RFC XXXX for
              valid combinations.

              If this leaf-list is not configured (has zero elements)
              the acceptable cipher suites are implementation-
              defined.";
```

```
          reference
           "RFC XXXX: YANG Groupings for TLS Clients and TLS Servers";
        }
      }

    } // end hello-params-grouping

  }
  <CODE ENDS>
```

6. Security Considerations

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

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

Since the modules defined in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

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

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

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

NONE

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

NONE

7. IANA Considerations

7.1. The IETF XML Registry

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

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

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

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

7.2. The YANG Module Names Registry

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

```
name:         ietf-tls-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-client
prefix:       tlsc
reference:    RFC XXXX

name:         ietf-tls-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-server
prefix:       tlss
reference:    RFC XXXX

name:         ietf-tls-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-common
prefix:       tlscmn
reference:    RFC XXXX
```

8. References

8.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-01 (work in progress), September 2018.

[I-D.ietf-netconf-keystore]
Watsen, K., "YANG Data Model for a Centralized Keystore Mechanism", draft-ietf-netconf-keystore-06 (work in progress), September 2018.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "YANG Data Model for Global Trust Anchors", draft-ietf-netconf-trust-anchors-01 (work in progress), September 2018.

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

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <https://www.rfc-editor.org/info/rfc5289>.

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

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

8.2. Informative References

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <https://www.rfc-editor.org/info/rfc2246>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

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

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

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

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Appendix A. Change Log

A.1. 00 to 01

- 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 verions 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 matricies to Section 5 (thanks Frank!)

- Claified that any configured "cipher-suite" values need to be compatible with the configured private key.

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
Juniper Networks

EMail: kwatsen@juniper.net

Gary Wu
Cisco Systems

EMail: garywu@cisco.com

Liang Xia
Huawei

EMail: frank.xialiang@huawei.com

NETCONF Working Group   K. Watsen
Internet-Draft   Juniper Networks
Intended status: Standards Track   October 22, 2018
Expires: April 25, 2019

YANG Data Model for Global Trust Anchors
draft-ietf-netconf-trust-anchors-02

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:

- "2018-10-22" --> 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

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 12
4. IANA Considerations 12
  4.1. The IETF XML Registry 12
  4.2. The YANG Module Names Registry 13
5. References 13
  5.1. Normative References 13
  5.2. Informative References 13
Appendix A. Change Log 15
  A.1. 00 to 01 15
  A.2. 01 to 02 15
Acknowledgements 15
Author's Address 15

1. Introduction

This document defines a YANG 1.1 [RFC7950] data model for configuring global sets of X.509 certificates and SSH host-keys that can be referenced by other data models for trust. While the SSH host-keys are uniquely for the SSH protocol, the X.509 certificates may be used for multiple uses, including authenticating protocol peers and verifying signatures.

This document in compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, to support trust anchors installed during manufacturing, it is expected that such data may appear only in <operational>.

1.1. Requirements Language

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

1.2. Tree Diagram Notation

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. The Trust Anchors Model

2.1. Tree Diagram

The following tree diagram provides an overview of the "ietf-trust-anchors" module.

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

2.2. Example Usage

The following example illustrates trust anchors in <operational> as described by Section 5.3 in [RFC8342]. This datastore view illustrates data set by the manufacturing process alongside conventional configuration. This trust anchors instance has six sets of pinned certificates and one set of pinned host keys.


```
   <trust-anchors
      xmlns="urn:ietf:params:xml:ns:yang:ietf-trust-anchors"
      xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">

     <!-- Manufacturer's trusted root CA certs -->
     <pinned-certificates or:origin="or:system">
       <name>manufacturers-root-ca-certs</name>
       <description>
         Certificates built into the device for authenticating
         manufacturer-signed objects, such as TLS server certificates,
         vouchers, etc.  Note, though listed here, these are not
         configurable; any attempt to do so will be denied.
       </description>
       <pinned-certificate>
         <name>Manufacturer Root CA cert 1</name>
         <cert>base64encodedvalue==</cert>
       </pinned-certificate>
       <pinned-certificate>
         <name>Manufacturer Root CA cert 2</name>
         <cert>base64encodedvalue==</cert>
       </pinned-certificate>
     </pinned-certificates>
```

```
<!-- specific end-entity certs for authenticating servers -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-server-certs</name>
  <description>
    Specific server authentication certificates for explicitly
    trusted servers.  These are needed for server certificates
    that are not signed by a pinned CA.
  </description>
  <pinned-certificate>
    <name>Fred Flintstone</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- trusted CA certs for authenticating servers -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-server-ca-certs</name>
  <description>
    Trust anchors (i.e. CA certs) that are used to authenticate
    server connections.  Servers are authenticated if their
    certificate has a chain of trust to one of these CA
    certificates.
  </description>
  <pinned-certificate>
    <name>ca.example.com</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- specific end-entity certs for authenticating clients -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-client-certs</name>
  <description>
    Specific client authentication certificates for explicitly
    trusted clients.  These are needed for client certificates
    that are not signed by a pinned CA.
  </description>
  <pinned-certificate>
    <name>George Jetson</name>
    <cert>base64encodedvalue==</cert>
  </pinned-certificate>
</pinned-certificates>

<!-- trusted CA certs for authenticating clients -->
<pinned-certificates or:origin="or:intended">
  <name>explicitly-trusted-client-ca-certs</name>
  <description>
    Trust anchors (i.e. CA certs) that are used to authenticate
```

```
         client connections.  Clients are authenticated if their
         certificate has a chain of trust to one of these CA
         certificates.
       </description>
       <pinned-certificate>
         <name>ca.example.com</name>
         <cert>base64encodedvalue==</cert>
       </pinned-certificate>
     </pinned-certificates>

     <!-- trusted CA certs for random HTTPS servers on Internet -->
     <pinned-certificates or:origin="or:system">
       <name>common-ca-certs</name>
       <description>
         Trusted certificates to authenticate common HTTPS servers.
         These certificates are similar to those that might be
         shipped with a web browser.
       </description>
       <pinned-certificate>
         <name>ex-certificate-authority</name>
         <cert>base64encodedvalue==</cert>
       </pinned-certificate>
     </pinned-certificates>

     <!-- specific SSH host keys for authenticating clients -->
     <pinned-host-keys or:origin="or:intended">
       <name>explicitly-trusted-ssh-host-keys</name>
       <description>
         Trusted SSH host keys used to authenticate SSH servers.
         These host keys would be analogous to those stored in
         a known_hosts file in OpenSSH.
       </description>
       <pinned-host-key>
         <name>corp-fw1</name>
         <host-key>base64encodedvalue==</host-key>
       </pinned-host-key>
     </pinned-host-keys>

   </trust-anchors>
```

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

[Note: '\' line wrapping for formatting only]


```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <trust-anchors
    xmlns="urn:ietf:params:xml:ns:yang:ietf-trust-anchors">
    <pinned-certificates>
      <name>explicitly-trusted-client-certs</name>
      <pinned-certificate>
        <name>George Jetson</name>
        <certificate-expiration>
          <expiration-date>2018-08-05T14:18:53-05:00</expiration-dat\
e>
        </certificate-expiration>
      </pinned-certificate>
    </pinned-certificates>
  </trust-anchors>
</notification>
```


2.3. YANG Module

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

```
  <CODE BEGINS> file "ietf-trust-anchors@2018-10-22.yang"
module ietf-trust-anchors {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-trust-anchors";
  prefix "ta";

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

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

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

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

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


     description
      "This module defines a data model for configuring global
       trust anchors used by other data models.  The data model
       enables the configuration of sets of trust anchors.
       This data model supports configuring trust anchors for
       both X.509 certificates and SSH host keys.

       Copyright (c) 2018 IETF Trust and the persons identified
       as authors of the code. All rights reserved.

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

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

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: YANG Data Model for Global Trust Anchors";
     }



     /************************************************************/
     /*   Typedefs for leafrefs to commonly referenced objects   */
     /************************************************************/

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

     feature ssh-host-keys {
```

```
       description
        "The 'ssh-host-keys' feature indicates that the server
         implements the /trust-anchors/pinned-host-keys subtree.";
     }


     /************************************************************/
     /*   Typedefs for leafrefs to commonly referenced objects   */
     /************************************************************/

     typedef pinned-certificates-ref {
       type leafref {
         path "/ta:trust-anchors/ta:pinned-certificates/ta:name";
         require-instance false;
       }
       description
         "This typedef enables importing modules to easily define a
          leafref to a 'pinned-certificates' object.  The require
          instance attribute is false to enable the referencing of
          pinned certificates that exist only in <operational>.";
       reference
         "RFC 8342: Network Management Datastore Architecture (NMDA)";
     }

     typedef pinned-host-keys-ref {
       type leafref {
         path "/ta:trust-anchors/ta:pinned-host-keys/ta:name";
         require-instance false;
       }
       description
         "This typedef enables importing modules to easily define a
          leafref to a 'pinned-host-keys' object.  The require
          instance attribute is false to enable the referencing of
          pinned host keys that exist only in <operational>.";
       reference
         "RFC 8342: Network Management Datastore Architecture (NMDA)";
     }


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

     container trust-anchors {
       nacm:default-deny-write;

       description
         "Contains sets of X.509 certificates and SSH host keys.";
```

```
       list pinned-certificates {
         if-feature "x509-certificates";
         key name;
         description
           "A list of pinned certificates.  These certificates can be
            used by a server to authenticate clients, or by a client
            to authenticate servers.  Each list of pinned certificates
            SHOULD be specific to a purpose, as the list as a whole
            may be referenced by other modules.  For instance, a
            RESTCONF server's configuration might use a specific list
            of pinned certificates for when authenticating RESTCONF
            client connections.";
         leaf name {
           type string;
           description
             "An arbitrary name for this list of pinned certificates.";
         }
         leaf description {
           type string;
           description
             "An arbitrary description for this list of pinned
              certificates.";
         }
         list pinned-certificate {
           key name;
           description
             "A pinned certificate.";
           leaf name {
             type string;
             description
               "An arbitrary name for this pinned certificate. The
                name must be unique across all lists of pinned
                certificates (not just this list) so that leafrefs
                from another module can resolve to unique values.";
           }
           uses ct:trust-anchor-cert-grouping {
             refine cert {
               mandatory true;
             }
           }
         }
       }

       list pinned-host-keys {
         if-feature "ssh-host-keys";
         key name;
         description
           "A list of pinned host keys.  These pinned host-keys can
```

```
            be used by clients to authenticate SSH servers.  Each
            list of pinned host keys SHOULD be specific to a purpose,
            so the list as a whole may be referenced by other modules.
            For instance, a NETCONF client's configuration might
            point to a specific list of pinned host keys for when
            authenticating specific SSH servers.";
         leaf name {
           type string;
           description
             "An arbitrary name for this list of pinned SSH
              host keys.";
         }
         leaf description {
           type string;
           description
             "An arbitrary description for this list of pinned SSH
              host keys.";
         }
         list pinned-host-key {
           key name;
           description
             "A pinned host key.";
           leaf name {
             type string;
             description
               "An arbitrary name for this pinned host-key. Must be
                unique across all lists of pinned host-keys (not just
                this list) so that a leafref to it from another module
                can resolve to unique values.";
           }
           leaf host-key {
             type ct:ssh-host-key;
             mandatory true;
             description
               "The binary public key data for this pinned host key.";
             reference
               "RFC YYYY: Common YANG Data Types for Cryptography";
           }
         }
       }
     }

   }
     <CODE ENDS>
```

3. Security Considerations

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

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

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

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

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments.

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

4. IANA Considerations

4.1. The IETF XML Registry

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

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

4.2. The YANG Module Names Registry

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

```
name:         ietf-trust-anchors
namespace:    urn:ietf:params:xml:ns:yang:ietf-trust-anchors
prefix:       ta
reference:    RFC XXXX
```

5. References

5.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-01 (work in progress), September 2018.

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

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

5.2. Informative References

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

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

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

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Change Log

A.1. 00 to 01

- o Added features "x509-certificates" and "ssh-host-keys".
- o Added nacm:default-deny-write to "trust-anchors" container.

A.2. 01 to 02

- o Switched "list pinned-certificate" to use the "trust-anchor-cert-grouping" from crypto-types. Effectively the same definition as before.

Acknowledgements

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

Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

NETCONF   G. Zheng
Internet-Draft   T. Zhou
Intended status: Standards Track   A. Clemm
Expires: April 22, 2019   Huawei
                                                        October 19, 2018

UDP based Publication Channel for Streaming Telemetry
draft-ietf-netconf-udp-pub-channel-04

Abstract

This document describes a UDP-based publication channel for streaming telemetry use to collect data from devices. A new shim header is proposed to facilitate the distributed data collection mechanism which directly pushes data from line cards to the collector. Because of the lightweight UDP encapsulation, higher frequency and better transit performance can be achieved.

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].

Table of Contents

1. Introduction 3
2. Terminologies 4
3. Solution Overview 4
4. Transport Mechanisms 5
4.1. Dynamic Subscription 6
4.2. Configured Subscription 7
5. UDP Transport for Publication Channel 8
5.1. Design Overview 8
5.2. Data Format of the UPC Message Header 9
5.3. Options 10
5.3.1. Reliability Option 10
5.3.2. Fragmentation Option 11
5.4. Data Encoding 12
6. Using DTLS to Secure UPC 12
6.1. Transport 13
6.2. Port Assignment 14
6.3. DTLS Session Initiation 14
6.4. Sending Data 14
6.5. Closure 15
7. Congestion Control 15
8. A YANG Data Model for Management of UPC 16
9. YANG Module 16
10. IANA Considerations 18
11. Security Considerations 19
12. Acknowledgements 19
13. References 19
13.1. Normative References 19
13.2. Informative References 20
13.3. URIs 21
Appendix A. Change Log 21
Authors' Addresses 22

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.

Sub-Notif [I-D.ietf-netconf-subscribed-notifications] defines a mechanism that allows a collector to subscribe to updates of YANG-defined data that is maintained in a YANG [RFC7950] datastore. The mechanism separates the management and control of subscriptions from the transport that is used to actually stream and deliver the data. Two transports, NETCONF transport [I-D.ietf-netconf-netconf-event-notifications] and HTTP transport [I-D.ietf-netconf-restconf-notif], have been defined so far for the notification messages.

While powerful in its features and general in its architecture, in its current form the mechanism needs to be extended to stream telemetry data at high velocity from devices that feature a distributed architecture. The transports that have been defined so far, NETCONF and HTTP, are ultimately based on TCP and lack the efficiency needed to stream data continuously at high velocity. A lighter-weight, more efficient transport, e.g. a transport based on UDP is needed.

- o Firstly, data collector will suffer a lot of TCP connections from, for example, many line cards equipped on different devices.
- o Secondly, as no connection state needs to be maintained, UDP encapsulation can be easily implemented by hardware which will further improve the performance.
- o Thirdly, because of the lightweight UDP encapsulation, higher frequency and better transit performance can be achieved, which is important for streaming telemetry.

This document specifies a higher-performance transport option for Sub-Notif that leverages UDP. Specifically, it facilitates the distributed data collection mechanism described in [I-D.zhou-netconf-multi-stream-originators]. In the case of data originating from multiple line cards, the centralized design requires data to be internally forwarded from those line cards to the push server, presumably on a main board, which then combines the individual data items into a single consolidated stream. The centralized data collection mechanism can result in a performance bottleneck, especially when large amounts of data are involved. What is needed instead is the support for a distributed mechanism that allows to directly push multiple individual substreams, e.g. one from each line card, 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. The proposed UDP based Publication Channel (UPC) natively supports the distributed data collection mechanism.

The transport described in this document can be used for transmitting notification messages over both IPv4 and IPv6 [RFC8200].

While this document will focus on the data publication channel, the subscription can be used in conjunction with the mechanism proposed in [I-D.ietf-netconf-subscribed-notifications] with extensions [I-D.zhou-netconf-multi-stream-originators].

2. Terminologies

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.

Component Subscription: A subscription that defines the data from each individual telemetry source which is managed and controlled by a single Subscription Server.

Component Subscription Server: An agent that streams telemetry data per the terms of a component subscription.

3. Solution Overview

The typical distributed data collection solution is shown in Fig. 1. Both the Collector and the Publisher can be distributed. The Collector includes the Subscriber and a set of Receivers. And the Publisher includes a Subscription Server and a set of Component Subscription Servers. The Subscriber cannot see the Component Subscription Servers directly, so it will send the Global Subscription information to the Subscription Server (e.g., main board) via the Subscription Channel. When receiving a Global Subscription, the Subscription Server decomposes the subscription request into multiple Component Subscriptions, each involving data from a separate internal telemetry source, for example a line card. The Component Subscriptions are distributed to the Component Subscription Server. Subsequently, each data originator generates its own stream of telemetry data, collecting and encapsulating the packets per the Component Subscription and streaming them to the designated Receivers. This distributed data collection mechanism may form multiple Publication Channels to the Receivers. The Receiver is able to assemble many pieces of data associated with one Global Subscription.

The Publication Channel supports the reliable data streaming, for example for some alarm events. The Collector has the option of deducing the packet loss and the disorder based on the information carried by the notification data. And the Collector may decide the behavior to request retransmission.

The rest of the draft describes the UDP based Publication Channel (UPC).

```
                 +--------------------------------------+
                 |            Collector                 |
                 |                                      |
                 |  +------------+  +-----------+       |
                 |  | Subscriber |  | Receivers |       |
                 |  +----+-------+  +--^----^---+       |
                 |       |             |    |           |
                 +--------------------------------------+
                         |             |    |
           Subscription |             |    | Publication
           Channel      |             |    | Channel
                         |   +---------+    |
                         |   |              |
                 +--------------------------------------+
                 |       |   |              |           |
                 |  +----v---+-----+ +------+-------+   |
                 |  | Subscription | | Component    |   |
                 |  | Server       | | Subscription |   |
                 |  |              | | Servers      |   |
                 |  +--------------+ +--------------+   |
                 |                                      |
                 |                Publisher             |
                 +--------------------------------------+
```

Fig. 1 Distributed Data Collection

4. Transport Mechanisms

For a complete pub-sub mechanism, this section will describe how the UPC is used to interact with the Subscription Channel relying on NETCONF or RESTCONF.

4.1. Dynamic Subscription

Dynamic subscriptions for Sub-Notif are configured and managed via signaling messages transported over NETCONF [RFC6241] or RESTCONF [RFC8040]. The Sub-Notif defined RPCs which are sent and responded via the Subscription Channel (a), between the Subscriber and the Subscription Server of the Publisher. In this case, only one Receiver is associated with the Subscriber. In the Publisher, there may be multiple data originators. Notification messages are pushed on separate channels (b), from different data originators to the Receiver.

```
+--------------+                         +--------------+
|  Collector   |                         |  Publisher   |
|              |                         |              |
|  (a)   (b)   |                         |  (a)    (b)  |
+--+------+----+                         +--+-------+---+
   |      |                                 |       |
   |      |     RPC:establish-subscription  |       |
   +---------------------------------------->       |
   |      |     RPC Reply: OK               |       |
   <----------------------------------------+       |
   |      |     UPC:notifications           |       |
   |      <-----------------------------------------+
   |      |                                 |       |
   |      |     RPC:modify-subscription     |       |
   +---------------------------------------->       |
   |      |     RPC Reply: OK               |       |
   <----------------------------------------+       |
   |      |     UPC:notifications           |       |
   |      <-----------------------------------------+
   |      |                                 |       |
   |      |     RPC:delete-subscription     |       |
   +---------------------------------------->       |
   |      |     RPC Reply: OK               |       |
   <----------------------------------------+       |
   |      |                                 |       |
   |      |                                 |       |
   +      +                                 +       +
```

Fig. 2 Call Flow For Dynamic Subscription

In the case of dynamic subscription, the Receiver and the Subscriber SHOULD be colocated. So UPC can use the source IP address of the Subscription Channel as it’s destination IP address. The Receiver MUST support listening messages at the IANA-assigned PORT-X or PORT-Y, but MAY be configured to listen at a different port.

For dynamic subscription, the Publication Channels MUST share fate with the subscription session. In other words, when the delete-subscription is received or the subscription session is broken, all the associated Publication Channels MUST be closed.

4.2. Configured Subscription

For a Configured Subscription, there is no guarantee that the Subscriber is currently in place with the associated Receiver(s). As defined in Sub-Notif, the subscription configuration contains the location information of all the receivers, including the IP address and the port number. So that the data originator can actively send generated messages to the corresponding Receivers via the UPC.

The first message MUST be a separate subscription-started notification to indicate the Receiver that the pushing is started. Then, the notifications can be sent immediately without any wait.

All the subscription state notifications, as defined in [I-D.ietf-netconf-subscribed-notifications], MUST be encapsulated to be separated notification messages.

```
   +---------------+                        +---------------+
   |  Collector    |                        |  Publisher    |
   |               |                        |               |
   |  (a)   (b)    |                        |  (a)    (b)   |
   +--+------+-----+                        +--+-------+----+
      |      |                                 |       |
      |      |     Capability Exchange         |       |
      <---------------------------------------->       |
      |      |                                 |       |
      |      |     Edit config(create)         |       |
      +---------------------------------------->       |
      |      |     RPC Reply: OK               |       |
      <----------------------------------------+       |
      |      |     UPC:subscription started    |       |
      |      <-----------------------------------------+
      |      |     UPC:notifications           |       |
      |      <-----------------------------------------+
      |      |                                 |       |
      |      |     Edit config(delete)         |       |
      +---------------------------------------->       |
      |      |     RPC Reply: OK               |       |
      <----------------------------------------+       |
      |      |     UPC:subscription terminated |       |
      |      <-----------------------------------------+
      |      |                                 |       |
      |      |                                 |       |
      +      +                                 +       +
```

Fig. 3 Call Flow For Configured Subscription

5. UDP Transport for Publication Channel

5.1. Design Overview

As specified in Sub-Notif, the telemetry data is encapsulated in the NETCONF/RESTCONF notification message, which is then encapsulated and carried in the transport protocols, e.g. TLS, HTTP2. The following figure shows the overview of the typical UPC message structure.

- o The Message Header contains information that can facilitate the message transmission before de-serializing the notification message.

- o Notification Message is the encoded content that the publication channel transports. The common encoding method includes GPB [1], CBOR [RFC7049], JSON, and XML. [I-D.ietf-netconf-notification-messages] describes the structure of the Notification Message for both single notification and multiple bundled notifications.

```
+-------+  +-------------+  +--------------+
|  UDP  |  |   Message   |  | Notification |
|       |  |   Header    |  | Message      |
+-------+  +-------------+  +--------------+
```

Fig. 4 UDP Publication Message Overview

5.2. Data Format of the UPC Message Header

The UPC Message Header contains information that can facilitate the message transmission before de-serializing the notification message. The data format is shown as follows.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-------+---------------+-------+-------------------------------+
 | Vers. |     Flag      |  ET   |       Length                  |
 +-------+---------------+-------+-------------------------------+
 |                       Message-Generator-ID                    |
 +---------------------------------------------------------------+
 |                       Message ID                              |
 +---------------------------------------------------------------+
 ~                       Options                                 ~
 +---------------------------------------------------------------+
```

Fig. 3 UPC Message Header Format

The Message Header contains the following field:

- o Vers.: represents the PDU (Protocol Data Unit) encoding version. The initial version value is 0.

- o Flag: is a bitmap indicating what features this packet has and the corresponding options attached. Each bit associates to one feature and one option data. When the bit is set to 1, the associated feature is enabled and the option data is attached. The sequence of the presence of the options follows the bit order of the bitmap. In this document, the flag is specified as follows:

  * bit 0, the reliability flag;

  * bit 1, the fragmentation flag;

  * other bits are reserved.

- ET: is a 4 bits identifier to indicate the encoding type used for the Notification Message. 16 types of encoding can be expressed:

  * 0: GPB;

  * 1: CBOR;

  * 2: JSON;

  * 3: XML;

  * others are reserved.

- Length: is the total length of the message, measured in octets, including message header.

- Message-Generator-ID: is a 32-bit identifier of the process which created the notification message. This allows disambiguation of an information source, such as the identification of different line cards sending the notification messages. The source IP address of the UDP datagrams SHOULD NOT be interpreted as the identifier for the host that originated the UPC message. The entity sending the UPC message could be merely a relay.

- The Message ID is generated continuously by the message generator. Different subscribers share the same notification ID sequence.

- Options: is a variable-length field. The details of the Options will be described in the respective sections below.

5.3. Options

The order of packing the data fields in the Options field follows the bit order of the Flag field.

5.3.1. Reliability Option

The UDP based publication transport described in this document provides two streaming modes, the reliable mode an the unreliable mode, for different SLA (Service Level Agreement) and telemetry requirements.

In the unreliable streaming mode, the line card pushes the encapsulated data to the data collector without any sequence information. So the subscriber does not know whether the data is correctly received or not. Hence no retransmission happens.

The reliable streaming mode provides sequence information in the UDP packet, based on which the subscriber can deduce the packet loss and disorder. Then the subscriber can decide whether to request the retransmission of the lost packets.

In most case, the unreliable streaming mode is preferred. Because the reliable streaming mode will cost more network bandwidth and precious device resource. Different from the unreliable streaming mode, the line card cannot remove the sent reliable notifications immediately, but to keep them in the memory for a while. Reliable notifications may be pushed multiple times, which will increase the traffic. When choosing the reliable streaming mode or the unreliable streaming mode, the operate need to consider the reliable requirement together with the resource usage.

When the reliability flag bit is set to 1 in the Flag field, the following option data will be attached

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+----------------------------------------------------------------+
|             Previous Message ID                                |
+----------------------------------------------------------------+
```

Fig. 4 Reliability Option Format

Current Message ID and Previous Message ID will be added in the packets.

For example, there are two subscriber A and B,

- o Message IDs for the generator are : [1, 2, 3, 4, 5, 6, 7, 8, 9], in which Subscriber A subscribes [1,2,3,6,7] and Subscriber B subscribes [1,2,4,5,7,8,9].
- o Subscriber A will receive [Previous Message ID, Current Message ID] like: [0,1][1,2][2,3][3,6][6,7].
- o Subscriber B will receive [Previous Message ID, Current Message ID] like: [0,1][1,2][2,4][4,5][5,7][7,8][8,9].

5.3.2. Fragmentation Option

UDP palyload has a theoretical length limitation to 65535. Other encapsulation headers will make the actual payload even shorter. Binary encodings like GPB and CBOR can make the message compact. So that the message can be encapsulated within one UDP packet, hence fragmentation will not easily happen. However, text encodings like

JSON and XML can easily make the message exceed the UDP length limitation.

The Fragmentation Option can help not Application layer can split the YANG tree into several leaves. Or table into several rows. But the leaf or the row cannot be split any further. Now we consider a very long path. Since the GPB and CBOR are so compact, it's easy to fit into a UDP packet. But for JSON or XML, it is possible that even one leaf will exceed the UDP boundary.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------------------------------------+-+
|            Fragment Number                                  |L|
+-------------------------------------------------------------+-+
```

Fig. 5 Fragmentation Option Format

The Fragmentation Option is available in the message header when the fragmentation flag is set to 1. The option contains:

Fragment Number: indicates the sequence number of the current fragment.

L: is a flag to indicate whether the current fragment is the last one. When 0 is set, current fragment is not the last one, hence more fragments are expected. When 1 is set, current fragment is the last one.

5.4. Data Encoding

Subscribed data can be encoded in GPB, CBOR, XML or JSON format. It is conceivable that additional encodings may be supported as options in the future. This can be accomplished by augmenting the subscription data model with additional identity statements used to refer to requested encodings.

Implementation may support different encoding method per subscription. When bundled notifications is supported between the publisher and the receiver, only subscribed notifications with the same encoding can be bundled as one message.

6. Using DTLS to Secure UPC

The Datagram Transport Layer Security (DTLS) protocol [RFC6347] is designed to meet the requirements of applications that need secure datagram transport.

DTLS can be used as a secure transport to counter all the primary threats to UDP based Publication Channel:

- o Confidentiality to counter disclosure of the message contents.

- o Integrity checking to counter modifications to a message on a hop-by-hop basis.

- o Server or mutual authentication to counter masquerade.

In addition, DTLS also provides:

- o A cookie exchange mechanism during handshake to counter Denial of Service attacks.

- o A sequence number in the header to counter replay attacks.

6.1. Transport

As shown in Figure 6, the DTLS is layered next to the UDP transport is to provide reusable security and authentication functions over UDP. No DTLS extension is required to enable UPC messages over DTLS.

```
+------------------------------+
|          UPC Message         |
+------------------------------+
|            DTLS              |
+------------------------------+
|            UDP               |
+------------------------------+
|            IP                |
+------------------------------+
```

Fig. 6: Protocol Stack for DTLS secured UPC

The application implementer will map a unique combination of the remote address, remote port number, local address, and local port number to a session.

Each UPC message is delivered by the DTLS record protocol, which assigns a sequence number to each DTLS record. Although the DTLS implementer may adopt a queue mechanism to resolve reordering, it may not assure that all the messages are delivered in order when mapping on the UDP transport.

Since UDP is an unreliable transport, with DTLS, an originator or relay may not realize that a collector has gone down or lost its DTLS connection state, so messages may be lost.

The DTLS record has its own sequence number, the encryption and decryption will done by DTLS layer, UPC Message layer will not concern this.

6.2. Port Assignment

The Publisher is always a DTLS client, and the Receiver is always a DTLS server. The Receivers MUST support accepting UPC Messages on the UDP port PORT-Y, but MAY be configurable to listen on a different port. The Publisher MUST support sending UPC messages to the UDP port PORT-Y, but MAY be configurable to send messages to a different port. The Publisher MAY use any source UDP port for transmitting messages.

6.3. DTLS Session Initiation

The Publisher initiates a DTLS connection by sending a DTLS Client Hello to the Receiver. Implementations MUST support the denial of service countermeasures defined by DTLS. When these countermeasures are used, the Receiver responds with a DTLS Hello Verify Request containing a cookie. The Publisher responds with a DTLS Client Hello containing the received cookie, which initiates the DTLS handshake. The Publisher MUST NOT send any UPC messages before the DTLS handshake has successfully completed.

Implementations MUST support DTLS 1.0 [RFC4347] and MUST support the mandatory to implement cipher suite, which is TLS_RSA_WITH_AES_128_CBC_SHA [RFC5246] as specified in DTLS 1.0. If additional cipher suites are supported, then implementations MUST NOT negotiate a cipher suite that employs NULL integrity or authentication algorithms.

Where privacy is REQUIRED, then implementations must either negotiate a cipher suite that employs a non-NULL encryption algorithm or else achieve privacy by other means, such as a physically secured network.

6.4. Sending Data

All UPC messages MUST be sent as DTLS "application_data". It is possible that multiple UPC messages be contained in one DTLS record, or that a publication message be transferred in multiple DTLS records. The application data is defined with the following ABNF [RFC5234] expression:

```
APPLICATION-DATA = 1*UPC-FRAME

UPC-FRAME = MSG-LEN SP UPC-MSG
```

MSG-LEN = NONZERO-DIGIT *DIGIT

SP = %d32

NONZERO-DIGIT = %d49-57

DIGIT = %d48 / NONZERO-DIGIT

UPC-MSG is defined in section 5.2.

6.5. Closure

A Publisher MUST close the associated DTLS connection if the connection is not expected to deliver any UPC Messages later. It MUST send a DTLS close_notify alert before closing the connection. A Publisher (DTLS client) MAY choose to not wait for the Receiver’s close_notify alert and simply close the DTLS connection. Once the Receiver gets a close_notify from the Publisher, it MUST reply with a close_notify.

When no data is received from a DTLS connection for a long time (where the application decides what "long" means), Receiver MAY close the connection. The Receiver (DTLS server) MUST attempt to initiate an exchange of close_notify alerts with the Publisher before closing the connection. Receivers that are unprepared to receive any more data MAY close the connection after sending the close_notify alert.

Although closure alerts are a component of TLS and so of DTLS, they, like all alerts, are not retransmitted by DTLS and so may be lost over an unreliable network.

7. Congestion Control

Congestion control mechanisms that respond to congestion by reducing traffic rates and establish a degree of fairness between flows that share the same path are vital to the stable operation of the Internet [RFC2914]. While efficient, UDP has no build-in congestion control mechanism. Because streaming telemetry can generate unlimited amounts of data, transferring this data over UDP is generally problematic. It is not recommended to use the UDP based publication channel over congestion-sensitive network paths. The only environments where the UDP based publication channel MAY be used are managed networks. The deployments require the network path has been explicitly provisioned for the UDP based publication channel through traffic engineering mechanisms, such as rate limiting or capacity reservations.

8. A YANG Data Model for Management of UPC

The YANG model defined in Section 9 has two leafs augmented into one place of Sub-Notif [I-D.ietf-netconf-subscribed-notifications], plus one identities.

```
  module: ietf-upc-subscribed-notifications
     augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
       +--rw address?   inet:ip-address
       +--rw port?      inet:port-number
```

9. YANG Module

```
<CODE BEGINS> file "ietf-upc-subscribed-notifications@2018-10-19.yang"
module ietf-upc-subscribed-notifications {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-upc-subscribed-notifications";
  prefix upcsn;
  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:   Guangying Zheng
               <mailto:zhengguangying@huawei.com>

     Editor:   Tianran Zhou
               <mailto:zhoutianran@huawei.com>

     Editor:   Alexander Clemm
               <mailto:alexander.clemm@huawei.com>";


  description
    "Defines UDP Publish Channel as a supported transport for subscribed
    event notifications.

    Copyright (c) 2018 IETF Trust and the persons identified as authors
    of the code.  All rights reserved.
```

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

     This version of this YANG module is part of RFC XXXX; see the RFC

     itself for full legal notices.";

   revision 2018-10-19 {
     description
       "Initial version";
     reference
       "RFC XXXX: UDP based Publication Channel for Streaming Telemetry";
   }

   identity upc {
     base sn:transport;
     description
       "UPC is used as transport for notification messages and state
        change notifications.";
   }

   grouping target-receiver {
     description
       "Provides a reusable description of a UPC target receiver.";
     leaf address {
       type inet:ip-address;
       description
         "Ip address of target upc receiver, which can be IPv4 address or
          IPV6 address.";
     }
     leaf port {
       type inet:port-number;
       description
         "Port number of target UPC receiver, if not specify, system
          should use default port number.";
     }
   }

   augment "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
     description
       "This augmentation allows UPC specific parameters to be
        exposed for a subscription.";
     uses target-receiver;
   }
 }
```

<CODE ENDS>

10. IANA Considerations

This RFC requests that IANA assigns three UDP port numbers in the "Registered Port Numbers" range with the service names "upc" and "upc-dtls". These ports will be the default ports for the UDP based Publication Channel for NETCONF and RESTCONF. Below is the registration template following the rules in [RFC6335].

Service Name: upc

Transport Protocol(s): UDP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: UDP based Publication Channel

Reference: RFC XXXX

Port Number: PORT-X

Service Name: upc-dtls

Transport Protocol(s): UDP

Assignee: IESG <iesg@ietf.org>

Contact: IETF Chair <chair@ietf.org>

Description: UDP based Publication Channel (DTLS)

Reference: RFC XXXX

Port Number: PORT-Y

IANA is requested to assign a new URI from the IETF XML Registry [RFC3688]. The following URI is suggested:

URI: urn:ietf:params:xml:ns:yang:ietf-upc-subscribed-notifications
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document also requests a new YANG module name in the YANG Module Names registry [RFC7950] with the following suggestion:

name: ietf-upc-subscribed-notifications
namespace: urn:ietf:params:xml:ns:yang:ietf-upc-subscribed-notifications
prefix: upcsn
reference: RFC XXXX

11. Security Considerations

TBD

12. Acknowledgements

The authors of this documents would like to thank Eric Voit, Tim Jenkins, and Huiyang Yang for the initial comments.

13. References

13.1. Normative References

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

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <https://www.rfc-editor.org/info/rfc2914>.

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

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, DOI 10.17487/RFC4347, April 2006, <https://www.rfc-editor.org/info/rfc4347>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

13.2. Informative References

[I-D.ietf-netconf-netconf-event-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "NETCONF Support for Event Notifications", draft-ietf-netconf-netconf-event-notifications-13 (work in progress), October 2018.

[I-D.ietf-netconf-notification-messages]
Voit, E., Birkholz, H., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-04 (work in progress), August 2018.

[I-D.ietf-netconf-restconf-notif]
Voit, E., Rahman, R., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "RESTCONF Transport for Event Notifications", draft-ietf-netconf-restconf-notif-08 (work in progress), October 2018.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-17 (work in progress), September 2018.

[I-D.zhou-netconf-multi-stream-originators]
Zhou, T., Zheng, G., Voit, E., Clemm, A., and A. Bierman, "Subscription to Multiple Stream Originators", draft-zhou-netconf-multi-stream-originators-03 (work in progress), October 2018.

13.3. URIs

[1] https://developers.google.com/protocol-buffers/

Appendix A. Change Log

(To be removed by RFC editor prior to publication)

A.1. draft-ietf-zheng-udp-pub-channel-00 to v00

- o Modified the message header format.
- o Added a section on the Authentication Option.
- o Cleaned up the text and removed unnecessary TBDs.

A.2. v01

- o Removed the detailed description on distributed data collection mechanism from this document. Mainly focused on the description of a UDP based publication channel for telemetry use.
- o Modified the message header format.

A.2. v02

- o Add the section on the transport mechanism.
- o Modified the fixed message header format.

o Add the fragmentation option for the message header.

A.2. v03

o Clarify term through the document.

o Add a section on DTLS support.

A.2. v04

o Add a section on UPC subscription model.

Authors' Addresses

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Alexander Clemm
Huawei
2330 Central Expressway
Santa Clara, California
USA

Email: alexander.clemm@huawei.com

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

A. Clemm
Huawei
E. Voit
Cisco Systems
A. Gonzalez Prieto
VMware
A. Tripathy
E. Nilsen-Nygaard
Cisco Systems
A. Bierman
YumaWorks
B. Lengyel
Ericsson
October 22, 2018

Subscription to YANG Datastores
draft-ietf-netconf-yang-push-20

Abstract

Via the mechanism described in this document, subscriber applications may request a continuous, customized stream of updates from a YANG datastore. Providing such visibility into updates enables new capabilities based on the remote mirroring and monitoring of configuration and operational state.

Copyright Notice

Table of Contents

1. Introduction 3
2. Definitions and Acronyms 4
3. Solution Overview 5
3.1. Subscription Model 5
3.2. Negotiation of Subscription Policies 6
3.3. On-Change Considerations 7
3.4. Reliability Considerations 8
3.5. Data Encodings 9
3.6. Defining the Selection with a Datastore 10
3.7. Streaming Updates 11
3.8. Subscription Management 13
3.9. Receiver Authorization 15
3.10. On-Change Notifiable Datastore Nodes 16
3.11. Other Considerations 17
4. A YANG Data Model for Management of Datastore Push Subscriptions 18
4.1. Overview 18
4.2. Subscription Configuration 23
4.3. YANG Notifications 24

4.4. YANG RPCs 25
5. YANG Module 30
6. IANA Considerations 47
7. Security Considerations 48
8. Acknowledgments 49
9. References 49
9.1. Normative References 49
9.2. Informative References 50
Appendix A. Appendix A: Subscription Errors 51
A.1. RPC Failures 51
A.2. Notifications of Failure 53
Appendix B. Changes Between Revisions 53
Authors' Addresses 57

1. Introduction

Traditional approaches to providing visibility into managed entities from remote have been built on polling. With polling, data is periodically requested and retrieved by a client from a server to stay up-to-date. However, there are issues associated with polling-based management:

- Polling incurs significant latency. This latency prohibits many application types.
- Polling cycles may be missed, requests may be delayed or get lost, often when the network is under stress and the need for the data is the greatest.
- Polling requests may undergo slight fluctuations, resulting in intervals of different lengths. The resulting data is difficult to calibrate and compare.
- For applications that monitor for changes, many remote polling cycles place unwanted and ultimately wasteful load on the network, devices, and applications, particularly when changes occur only infrequently.

A more effective alternative to polling is for an application to receive automatic and continuous updates from a targeted subset of a datastore. Accordingly, there is a need for a service that allows applications to subscribe to updates from a datastore and that enables the server (also referred to as publisher) to push and in effect stream those updates. The requirements for such a service have been documented in [RFC7923].

This document defines a corresponding solution that is built on top of "Custom Subscription to Event Streams"

[I-D.draft-ietf-netconf-subscribed-notifications]. Supplementing that work are YANG data model augmentations, extended RPCs, and new datastore specific update notifications. Transport options for [I-D.draft-ietf-netconf-subscribed-notifications] will work seamlessly with this solution.

2. Definitions and Acronyms

This document uses the terminology defined in [RFC7950], [RFC8341], [RFC8342], and [I-D.draft-ietf-netconf-subscribed-notifications]. In addition, the following terms are introduced:

- Datastore node: A node in the instantiated YANG data tree associated with a datastore. In this document, datastore nodes are often also simply referred to as "objects"

- Datastore node update: A data item containing the current value of a datastore node at the time the datastore node update was created, as well as the path to the datastore node.

- Datastore subscription: A subscription to a stream of datastore node updates.

- Datastore subtree: A datastore node and all its descendant datastore nodes

- On-change subscription: A datastore subscription with updates that are triggered when changes in subscribed datastore nodes are detected.

- Periodic subscription: A datastore subscription with updates that are triggered periodically according to some time interval.

- Selection filter: Evaluation and/or selection criteria, which may be applied against a targeted set of objects.

- Update record: A representation of one or more datastore node updates. In addition, an update record may contain which type of update led to the datastore node update (e.g., whether the datastore node was added, changed, deleted). Also included in the update record may be other metadata, such as a subscription id of the subscription as part of which the update record was generated. In this document, update records are often also simply referred to as "updates".

- Update trigger: A mechanism that determines when an update record needs to be generated.

   o YANG-Push: The subscription and push mechanism for datastore updates that is specified in this document.

3. Solution Overview

This document specifies a solution that provides a subscription service for updates from a datastore. This solution supports dynamic as well as configured subscriptions to updates of datastore nodes. Subscriptions specify when notification messages (also referred to as "push updates") should be sent and what data to include in update records. Datastore node updates are subsequently pushed from the publisher to the receiver per the terms of the subscription.

3.1. Subscription Model

YANG-push subscriptions are defined using a YANG data model. This model enhances the subscription model defined in [I-D.draft-ietf-netconf-subscribed-notifications] with capabilities that allow subscribers to subscribe to datastore node updates, specifically to specify the update triggers defining when to generate update records as well as what to include in an update record. Key enhancements include:

- o Specification of selection filters which identify targeted YANG datastore nodes and/or datastore subtrees for which updates are to be pushed.

- o Specification of update policies contain conditions which trigger the generation and pushing of new update records. There are two types of subscriptions, distinguished by how updates are triggered: periodic and on-change.

  - * For periodic subscriptions, the update trigger is specified by two parameters that define when updates are to be pushed. These parameters are the period interval with which to report updates, and an "anchor time", i.e. a reference point in time that can be used to calculate at which points in time periodic updates need to be assembled and sent.

  - * For on-change subscriptions, an update trigger occurs whenever a change in the subscribed information is detected. Included are additional parameters that include:

    - + Dampening period: In an on-change subscription, detected object changes should be sent as quickly as possible. However it may be undesirable to send a rapid series of object changes. Such behavior has the potential to exhaust resources in the publisher or receiver. In order to protect against that, a dampening period MAY be used to specify the interval which has to pass before successive update records for the same subscription are generated for a receiver. The dampening period collectively applies to the set of all datastore nodes selected by a single subscription. This means that when there is a change to one or more subscribed objects, an update record containing those objects is created immediately (when no dampening period is in effect) or at the end of a dampening period (when a dampening period is in fact in effect). If multiple changes to a single object occur during a dampening period, only the value that is in effect at the time when the update record is created is included. The dampening period goes into effect every time an update record completes assembly.

+ Change type: This parameter can be used to reduce the types of datastore changes for which updates are sent (e.g., you might only send an update when an object is created or deleted, but not when an object value changes).

+ Sync on start: defines whether or not a complete push-update of all subscribed data will be sent at the beginning of a subscription. Such early synchronization establishes the frame of reference for subsequent updates.

o An encoding (using anydata) for the contents of periodic and on-change push updates.

3.2. Negotiation of Subscription Policies

A dynamic subscription request SHOULD be declined if a publisher's assessment is that it may be unable to provide update records meeting the terms of an "establish-subscription" or "modify-subscription" RPC request. In this case, a subscriber may quickly follow up with a new RPC request using different parameters.

Random guessing of different parameters by a subscriber is to be discouraged. Therefore, in order to minimize the number of subscription iterations between subscriber and publisher, a dynamic subscription supports a simple negotiation between subscribers and publishers for subscription parameters. This negotiation is in the form of supplemental information which should be inserted within error responses to a failed RPC request. This returned error response information, when considered, should increase the likelihood of success for subsequent RPC requests. Such hints include suggested periodic time intervals, acceptable dampening periods, and size estimates for the number or objects which would be returned from a proposed selection filter. However, there are no guarantees that subsequent requests which consider these hints will be accepted.

3.3. On-Change Considerations

On-change subscriptions allow receivers to receive updates whenever changes to targeted objects occur. As such, on-change subscriptions are particularly effective for data that changes infrequently, yet for which applications need to be quickly notified whenever a change does occur with minimal delay.

On-change subscriptions tend to be more difficult to implement than periodic subscriptions. Accordingly, on-change subscriptions may not be supported by all implementations or for every object.

Whether or not to accept or reject on-change subscription requests when the scope of the subscription contains objects for which on-change is not supported is up to the publisher implementation. A publisher MAY accept an on-change subscription even when the scope of the subscription contains objects for which on-change is not supported. In that case, updates are sent only for those objects within the scope that do support on-change updates, whereas other objects are excluded from update records, even if their values change. In order for a subscriber to determine whether objects support on-change subscriptions, objects are marked accordingly on a publisher. Accordingly, when subscribing, it is the responsibility of the subscriber to ensure it is aware of which objects support on-change and which do not. For more on how objects are so marked, see Section 3.10.

Alternatively, a publisher MAY decide to simply reject an on-change subscription in case the scope of the subscription contains objects for which on-change is not supported. In case of a configured subscription, the publisher MAY suspend the subscription.

To avoid flooding receivers with repeated updates for subscriptions containing fast-changing objects, or objects with oscillating values, an on-change subscription allows for the definition of a dampening period. Once an update record for a given object is generated, no other updates for this particular subscription will be created until the end of the dampening period. Values sent at the end of the dampening period are the values that are current at the end of the dampening period of all changed objects. Changed objects include those which were deleted or newly created during that dampening period. If an object has returned to its original value (or even has been created and then deleted) during the dampening-period, that value (and not the interim change) will still be sent. This will indicate churn is occurring on that object.

On-change subscriptions can be refined to let users subscribe only to certain types of changes. For example, a subscriber might only want object creations and deletions, but not modifications of object values.

Putting it all together, following is the conceptual process for creating an update record as part of an on-change subscription:

1. Just before a change, or at the start of a dampening period, evaluate any filtering and any access control rules to ensure receiver is authorized to view all subscribed datastore nodes (filtering out any nodes for which this is not the case). The result is a set "A" of datastore nodes and subtrees.

2. Just after a change, or at the end of a dampening period, evaluate any filtering and any (possibly new) access control rules. The result is a set "B" of datastore nodes and subtrees.

3. Construct an update record, which takes the form of YANG patch record [RFC8072] for going from A to B.

4. If there were any changes made between A and B which canceled each other out, insert into the YANG patch record the last change made, even if the new value is no different from the original value (since changes that were made in the interim were canceled out). In case the changes involve creating a new datastore node, then deleting it, the YANG patch record will indicate deletion of the datastore node. Similarly, in case the changes involve deleting a new datastore node, then recreating it, the YANG patch record will indicate creation of the datastore node.

5. If the resulting patch record is non-empty, send it to the receiver.

Note: In cases where a subscriber wants to have separate dampening periods for different objects, the subscriber has the option to create multiple subscriptions with different selection filters.

3.4. Reliability Considerations

A subscription to updates from a datastore is intended to obviate the need for polling. However, in order to do so, it is critical that subscribers can rely on the subscription and have confidence that they will indeed receive the subscribed updates without having to worry about updates being silently dropped. In other words, a subscription constitutes a promise on the side of the publisher to provide the receivers with updates per the terms of the subscription.

Now, there are many reasons why a publisher may at some point no longer be able to fulfill the terms of the subscription, even if the subscription had been entered into with good faith. For example, the volume of datastore nodes may be larger than anticipated, the interval may prove too short to send full updates in rapid succession, or an internal problem may prevent objects from being collected. For this reason, the solution that is defined in this document mandates that a publisher notifies receivers immediately and reliably whenever it encounters a situation in which it is unable to keep the terms of the subscription, and provides the publisher with the option to suspend the subscription in such a case. This includes indicating the fact that an update is incomplete as part of a push-update or push-change-update notification, as well as emitting a subscription-suspended notification as applicable. This is described further in Section 3.11.1.

A publisher SHOULD reject a request for a subscription if it is unlikely that the publisher will be able to fulfill the terms of that subscription request. In such cases, it is preferable to have a subscriber request a less resource intensive subscription than to deal with frequently degraded behavior.

3.5. Data Encodings

3.5.1. Periodic Subscriptions

In a periodic subscription, the data included as part of an update record corresponds to data that could have been read using a retrieval operation.

3.5.2. On-Change Subscriptions

In an on-change subscription, update records need to indicate not only values of changed datastore nodes but also the types of changes that occurred since the last update. Therefore, encoding rules for data in on-change updates will generally follow YANG-patch operation as specified in [RFC8072]. The YANG-patch will describe what needs to be applied to the earlier state reported by the preceding update, to result in the now-current state. Note that contrary to [RFC8072], objects encapsulated are not restricted to only configuration objects.

A publisher indicates the type of change to a datastore node using the different YANG patch operations: the "create" operation is used for newly created objects (except entries in a user-ordered list), the "delete" operation is used for deleted objects (including in user-ordered lists), the "replace" operation is used when only the object value changes, the "insert" operation is used when a new entry is inserted in a list, and the "move" operation is used when an existing entry in a user-ordered list is moved.

However, a patch must be able to do more than just describe the delta from the previous state to the current state. As per Section 3.3, it must also be able to identify whether transient changes have occurred on an object during a dampening period. To support this, it is valid to encode a YANG patch operation so that its application would result in no change between the previous and current state. This indicates that some churn has occurred on the object. An example of this would be a patch that indicates a "create" operation for a datastore node where the receiver believes one already exists, or a "replace" operation which replaces a previous value with the same value. Note that this means that the "create" and "delete" errors described in [RFC8072] section 2.5 are not errors, and are valid operations with YANG-Push.

3.6. Defining the Selection with a Datastore

A subscription must specify both the selection filters and the datastore against which these selection filters will be applied. This information is used to choose and subsequently push data from the publisher's datastore to the receivers.

Only a single selection filter can be applied to a subscription at a time. An RPC request proposing a new selection filter replaces any existing filter. The following selection filter types are included in the yang-push data model, and may be applied against a datastore:

- subtree: A subtree selection filter identifies one or more datastore subtrees. When specified, update records will only come from the datastore nodes of selected datastore subtree(s). The syntax and semantics correspond to that specified for [RFC6241] section 6.

- xpath: An "xpath" selection filter is an XPath expression that returns a node set. When specified, updates will only come from the selected datastore nodes.

These filters are intended to be used as selectors that define which objects are within the scope of a subscription. A publisher MUST support at least one type of selection filter.

XPath itself provides powerful filtering constructs and care must be used in filter definition. Consider an XPath filter which only passes a datastore node when an interface is up. It is up to the receiver to understand implications of the presence or absence of objects in each update.

When the set of selection filtering criteria is applied for a periodic subscription, then they are applied whenever a periodic update record is constructed, and only datastore nodes that pass the filter and to which a receiver has access are provided to that receiver. If the same filtering criteria is applied to an on-change subscription, only the subset of those datastore nodes supporting on-change is provided. A datastore node which doesn't support on-change is never sent as part of an on-change subscription's "push-update" or "push-change-update" (see Section 3.7).

3.7. Streaming Updates

Contrary to traditional data retrieval requests, datastore subscription enables an unbounded series of update records to be streamed over time. Two generic YANG notifications for update records have been defined for this: "push-update" and "push-change-update".

A "push-update" notification defines a complete, filtered update of the datastore per the terms of a subscription. This type of YANG notification is used for continuous updates of periodic subscriptions. A "push-update" notification can also be used for the on-change subscriptions in two cases. First, it MUST be used as the initial "push-update" if there is a need to synchronize the receiver at the start of a new subscription. It also MAY be sent if the publisher later chooses to resync an on-change subscription. The "push-update" update record contains an instantiated datastore subtree with all of the subscribed contents. The content of the update record is equivalent to the contents that would be obtained had the same data been explicitly retrieved using a datastore retrieval operation using the same transport with the same filters applied.

A "push-change-update" notification is the most common type of update for on-change subscriptions. The update record in this case contains the set of changes that datastore nodes have undergone since the last notification message. In other words, this indicates which datastore nodes have been created, deleted, or have had changes to their values. In cases where multiple changes have occurred over the course of a dampening period and the object has not been deleted, the object's most current value is reported. (In other words, for each object, only one change is reported, not its entire history. Doing so would defeat the purpose of the dampening period.)

"Push-update" and "push-change-update" are encoded and placed within notification messages, and ultimately queued for egress over the specified transport.

The following is an example of a notification message for a subscription tracking the operational status of a single Ethernet interface (per [RFC8343]). This notification message is encoded XML over NETCONF as per [I-D.draft-ietf-netconf-netconf-event-notifications].

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>2017-10-25T08:00:11.22Z</eventTime>
 <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
   <id>1011</id>
   <datastore-contents>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
       <interface>
         <name>eth0</name>
         <oper-status>up</oper-status>
       </interface>
     </interfaces>
   </datastore-contents>
 </push-update>
</notification>
```

Figure 1: Push example

The following is an example of an on-change notification message for the same subscription.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>2017-10-25T08:22:33.44Z</eventTime>
 <push-change-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
   <id>89</id>
   <datastore-changes>
     <yang-patch>
       <patch-id>0</patch-id>
       <edit>
         <edit-id>edit1</edit-id>
         <operation>replace</operation>
         <target>/ietf-interfaces:interfaces</target>
         <value>
           <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
             <interface>
               <name>eth0</name>
               <oper-status>down</oper-status>
             </interface>
           </interfaces>
         </value>
       </edit>
     </yang-patch>
   </datastore-changes>
 </push-change-update>
</notification>
```


Figure 2: Push example for on change

Of note in the above example is the 'patch-id' with a value of '0'. Per [RFC8072], the 'patch-id' is an arbitrary string. With YANG Push, the publisher SHOULD put into the 'patch-id' a counter starting at '0' which increments with every 'push-change-update' generated for a subscription. If used as a counter, this counter MUST be reset to '0' anytime a resynchronization occurs (i.e., with the sending of a 'push-update'). Also if used as a counter, the counter MUST be reset to '0' after passing a maximum value of '4294967295' (i.e. maximum value that can be represented using uint32 data type). Such a mechanism allows easy identification of lost or out-of-sequence update records.

3.8. Subscription Management

The RPCs defined within [I-D.draft-ietf-netconf-subscribed-notifications] have been enhanced to support datastore subscription negotiation. Also, new error codes have been added that are able to indicate why a datastore subscription attempt has failed, along with new yang-data that MAY be used to include details on input parameters that might result in a successful subsequent RPC invocation.

The establishment or modification of a datastore subscription can be rejected for multiple reasons. This includes a too large subtree request, or the inability of the publisher to push update records as frequently as requested. In such cases, no subscription is established. Instead, the subscription-result with the failure reason is returned as part of the RPC response. As part of this response, a set of alternative subscription parameters MAY be returned that would likely have resulted in acceptance of the subscription request. The subscriber may consider these as part of future subscription attempts.

In the case of a rejected request for an establishment of a datastore subscription, if there are hints, the hints SHOULD be transported within a yang-data "establish-subscription-datastore-error-info" container inserted into the RPC error response, in lieu of the "establish-subscription-stream-error-info" that is inserted in case of a stream subscription.

Below is a tree diagram for "establish-subscription-datastore-error-info". All tree diagrams used in this document follow the notation defined in [RFC8340]

```
yang-data establish-subscription-datastore-error-info
   +--ro establish-subscription-datastore-error-info
      +--ro reason?                 identityref
      +--ro period-hint?            yang:timeticks
      +--ro filter-failure-hint?    string
      +--ro object-count-estimate? uint32
      +--ro object-count-limit?     uint32
      +--ro kilobytes-estimate?     uint32
      +--ro kilobytes-limit?        uint32
```

Figure 3: Tree diagram for establish-subscription-datastore-error-info

Similarly, in the case of a rejected request for modification of a datastore subscription, if there are hints, the hints SHOULD be transported within a yang-data "modify-subscription-datastore-error-info" container inserted into the RPC error response, in lieu of the "modify-subscription-stream-error-info" that is inserted in case of a stream subscription.

Below is a tree diagram for "modify-subscription-datastore-error-info".

```
          yang-data modify-subscription-datastore-error-info
             +--ro modify-subscription-datasore-error-info
                +--ro reason?                 identityref
                +--ro period-hint?            yang:timeticks
                +--ro filter-failure-hint?    string
                +--ro object-count-estimate? uint32
                +--ro object-count-limit?    uint32
                +--ro kilobytes-estimate?    uint32
                +--ro kilobytes-limit?       uint32
```

Figure 4: Tree diagram for modify-subscription-datastore-error-info

3.9. Receiver Authorization

A receiver of subscription data MUST only be sent updates for which it has proper authorization. A publisher MUST ensure that no non-authorized data is included in push updates. To do so, it needs to apply all corresponding checks applicable at the time of a specific pushed update and if necessary silently remove any non-authorized data from datastore subtrees. This enables YANG data pushed based on subscriptions to be authorized equivalently to a regular data retrieval (get) operation.

Each "push-update" and "push-change-update" MUST have access control applied, as is depicted in the following diagram. This includes validating that read access is permitted for any new objects selected since the last notification message was sent to a particular receiver. To accomplish this, implementations SHOULD support the conceptual authorization model of [RFC8341], specifically section 3.2.4.

```
                       +-----------------+      +--------------------+
  push-update or -->   | datastore node  | yes  | add datastore node |
 push-change-update    | access allowed? | ---> | to update record   |
                       +-----------------+      +--------------------+
```

Figure 5: Updated [RFC8341] access control for push updates

A publisher MUST allow for the possibility that a subscription's selection filter references non-existent data or data that a receiver is not allowed to access. Such support permits a receiver the ability to monitor the entire lifecyle of some datastore tree without needing to explicitly enumerate every individual datastore node. If, after access control has been applied, there are no objects remaining in an update record, then (in case of a periodic subscription) only a single empty "push-update" notification MUST be sent. Empty "push-change-update" messages (in case of an on-change subscription) MUST NOT be sent. This is required to ensure that clients cannot surreptitiously monitor objects that they do not have access to via carefully crafted selection filters. By the same token, changes to objects that are filtered MUST NOT affect any dampening intervals.

A publisher MAY choose to reject an establish-subscription request which selects non-existent data or data that a receiver is not allowed to access. As reason, the error identity "unchanging-selection" SHOULD be returned. In addition, a publisher MAY choose to terminate a dynamic subscription or suspend a configured receiver when the authorization privileges of a receiver change, or the access controls for subscribed objects change. In that case, the publisher SHOULD include the error identity "unchanging-selection" as reason when sending the "subscription-terminated" respectively "subscription-suspended" notification. Such a capability enables the publisher to avoid having to support continuous and total filtering of a subscription's content for every update record. It also reduces the possibility of leakage of access-controlled objects.

If read access into previously accessible nodes has been lost due to a receiver permissions change, this SHOULD be reported as a patch "delete" operation for on-change subscriptions. If not capable of handling such receiver permission changes with such a "delete", publisher implementations MUST force dynamic subscription re-establishment or configured subscription re-initialization so that appropriate filtering is installed.

3.10. On-Change Notifiable Datastore Nodes

In some cases, a publisher supporting on-change notifications may not be able to push on-change updates for some object types. Reasons for this might be that the value of the datastore node changes frequently (e.g., [RFC8343]'s 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 they are not supported. Otherwise client applications will have no way of knowing whether they can indeed rely on their on-change subscription to provide them with the change updates that they are interested in. In other words, if implementations do not provide a solution and do not support comprehensive on-change notifiability, clients of those implementations will have no way of knowing what their on-change subscription actually covers.

Implementations are therefore strongly advised to provide a solution to this problem. It is expected that such a solution will be standardized at some point in the future. In the meantime and until this occurs, implementations SHOULD provide their own solution.

3.11. Other Considerations

3.11.1. Robustness and reliability

Particularly in the case of on-change updates, it is important that these updates do not get lost. In case the loss of an update is unavoidable, it is critical that the receiver is notified accordingly.

Update records for a single subscription MUST NOT be resequenced prior to transport.

It is conceivable that under certain circumstances, a publisher will recognize that it is unable to include within an update record the full set of objects desired per the terms of a subscription. In this case, the publisher MUST act as follows.

- o The publisher MUST set the "incomplete-update" flag on any update record which is known to be missing information.

- o The publisher MAY choose to suspend the subscription as per [I-D.draft-ietf-netconf-subscribed-notifications]. If the publisher does not create an update record at all, it MUST suspend the subscription.

- o When resuming an on-change subscription, the publisher SHOULD generate a complete patch from the previous update record. If this is not possible and the "sync-on-start" option is true for the subscription, then the full datastore contents MAY be sent via a "push-update" instead (effectively replacing the previous contents). If neither of these are possible, then an "incomplete-update" flag MUST be included on the next "push-change-update".

Note: It is perfectly acceptable to have a series of "push-change-update" notifications (and even "push update" notifications) serially queued at the transport layer awaiting transmission. It is not required for the publisher to merge pending update records sent at the same time.

3.11.2. Publisher capacity

It is far preferable to decline a subscription request than to accept such a request when it cannot be met.

Whether or not a subscription can be supported will be determined by a combination of several factors such as the subscription update trigger (on-change or periodic), the period in which to report changes (one second periods will consume more resources than one hour periods), the amount of data in the datastore subtree that is being subscribed to, and the number and combination of other subscriptions that are concurrently being serviced.

4. A YANG Data Model for Management of Datastore Push Subscriptions

4.1. Overview

The YANG data model for datastore push subscriptions is depicted in the following figure. The tree diagram follows the notation defined in [RFC8340]. New schema objects defined here (i.e., beyond those from [I-D.draft-ietf-netconf-subscribed-notifications]) are identified with "yp". For the reader's convenience, in order to compact the tree representation, some nodes that are defined in ietf-subscribed-notifications and that are not essential to the understanding of the data model defined here have been removed. This is indicated by "..." in the diagram where applicable.

```
module: ietf-subscribed-notifications
    ...
    +--rw filters
    |  ...
    |  +--rw yp:selection-filter* [filter-id]
    |     +--rw yp:filter-id                   string
    |     +--rw (yp:filter-spec)?
    |        +--:(yp:datastore-subtree-filter)
    |        |  +--rw yp:datastore-subtree-filter?   <anydata>
    |        |          {sn:subtree}?
    |        +--:(yp:datastore-xpath-filter)
    |           +--rw yp:datastore-xpath-filter?     yang:xpath1.0
    |                   {sn:xpath}?
    +--rw subscriptions
       +--rw subscription* [id]
          |  ...
          +--rw (target)
          |  +--:(stream)
          |  |   ...
          |  +--:(yp:datastore)
          |     +--rw yp:datastore                          identityref
```

```
           |     +--rw (yp:selection-filter)?
           |        +--:(yp:by-reference)
           |        |  +--rw yp:selection-filter-ref
           |        |          selection-filter-ref
           |        +--:(yp:within-subscription)
           |           +--rw (yp:filter-spec)?
           |              +--:(yp:datastore-subtree-filter)
           |              |  +--rw yp:datastore-subtree-filter?
           |              |          <anydata> {sn:subtree}?
           |              +--:(yp:datastore-xpath-filter)
           |                 +--rw yp:datastore-xpath-filter?
           |                         yang:xpath1.0 {sn:xpath}?
           | ...
           +--rw (yp:update-trigger)
              +--:(yp:periodic)
              |  +--rw yp:periodic!
              |     +--rw yp:period         yang:timeticks
              |     +--rw yp:anchor-time?   yang:date-and-time
              +--:(yp:on-change) {on-change}?
                 +--rw yp:on-change!
                    +--rw yp:dampening-period?   yang:timeticks
                    +--rw yp:sync-on-start?      boolean
                    +--rw yp:excluded-change*    change-type

  rpcs:
    +---x establish-subscription
    |  +---w input
    |  |  ...
    |  |  +---w (target)
    |  |  |  +--:(stream)
    |  |  |  |  ...
    |  |  |  +--:(yp:datastore)
    |  |  |     +---w yp:datastore                   identityref
    |  |  |     +---w (yp:selection-filter)?
    |  |  |        +--:(yp:by-reference)
    |  |  |        |  +---w yp:selection-filter-ref
    |  |  |        |          selection-filter-ref
    |  |  |        +--:(yp:within-subscription)
    |  |  |           +---w (yp:filter-spec)?
    |  |  |              +--:(yp:datastore-subtree-filter)
    |  |  |              |  +---w yp:datastore-subtree-filter?
    |  |  |              |          <anydata> {sn:subtree}?
    |  |  |              +--:(yp:datastore-xpath-filter)
    |  |  |                 +---w yp:datastore-xpath-filter?
    |  |  |                         yang:xpath1.0 {sn:xpath}?
    |  |  | ...
    |  |  +---w (yp:update-trigger)
    |  |     +--:(yp:periodic)
```

```
     |  |     |  +---w yp:periodic!
     |  |     |     +---w yp:period         yang:timeticks
     |  |     |     +---w yp:anchor-time?   yang:date-and-time
     |  |     +--:(yp:on-change) {on-change}?
     |  |        +---w yp:on-change!
     |  |           +---w yp:dampening-period?   yang:timeticks
     |  |           +---w yp:sync-on-start?      boolean
     |  |           +---w yp:excluded-change*    change-type
     |  +--ro output
     |     +--ro id                            subscription-id
     |     +--ro replay-start-time-revision?   yang:date-and-time
     |             {replay}?
     +---x modify-subscription
     |  +---w input
     |     ...
     |     +---w (target)
     |     |  ...
     |     |  +--:(yp:datastore)
     |     |     +---w (yp:selection-filter)?
     |     |        +--:(yp:by-reference)
     |     |        |  +---w yp:selection-filter-ref
     |     |        |          selection-filter-ref
     |     |        +--:(yp:within-subscription)
     |     |           +---w (yp:filter-spec)?
     |     |              +--:(yp:datastore-subtree-filter)
     |     |              |  +---w yp:datastore-subtree-filter?
     |     |              |          <anydata> {sn:subtree}?
     |     |              +--:(yp:datastore-xpath-filter)
     |     |                 +---w yp:datastore-xpath-filter?
     |     |                         yang:xpath1.0 {sn:xpath}?
     |     | ...
     |     +---w (yp:update-trigger)
     |        +--:(yp:periodic)
     |        |  +---w yp:periodic!
     |        |     +---w yp:period         yang:timeticks
     |        |     +---w yp:anchor-time?   yang:date-and-time
     |        +--:(yp:on-change) {on-change}?
     |           +---w yp:on-change!
     |              +---w yp:dampening-period?   yang:timeticks
     +---x delete-subscription
     |  ...
     +---x kill-subscription
        ...

   yang-data (for placement into rpc error responses)
     ...

   notifications:
```

```
       +---n replay-completed {replay}?
       |  ...
       +---n subscription-completed
       |  ...
       +---n subscription-started {configured}?
       |  |  ...
       |  +--ro (target)
       |  |  ...
       |  |  +--:(yp:datastore)
       |  |     +--ro yp:datastore                   identityref
       |  |     +--ro (yp:selection-filter)?
       |  |        +--:(yp:by-reference)
       |  |        |  +--ro yp:selection-filter-ref
       |  |        |          selection-filter-ref
       |  |        +--:(yp:within-subscription)
       |  |           +--ro (yp:filter-spec)?
       |  |              +--:(yp:datastore-subtree-filter)
       |  |              |  +--ro yp:datastore-subtree-filter?
       |  |              |          <anydata> {sn:subtree}?
       |  |              +--:(yp:datastore-xpath-filter)
       |  |                 +--ro yp:datastore-xpath-filter?
       |  |                         yang:xpath1.0 {sn:xpath}?
       |  ...
       |  +--ro (yp:update-trigger)
       |     +--:(yp:periodic)
       |     |  +--ro yp:periodic!
       |     |     +--ro yp:period         yang:timeticks
       |     |     +--ro yp:anchor-time?   yang:date-and-time
       |     +--:(yp:on-change) {on-change}?
       |        +--ro yp:on-change!
       |           +--ro yp:dampening-period?   yang:timeticks
       |           +--ro yp:sync-on-start?      boolean
       |           +--ro yp:excluded-change*    change-type
       +---n subscription-resumed
       |  ...
       +---n subscription-modified
       |  ...
       |  +--ro (target)
       |  |  |  ...
       |  |  +--:(yp:datastore)
       |  |     +--ro yp:datastore                   identityref
       |  |     +--ro (yp:selection-filter)?
       |  |        +--:(yp:by-reference)
       |  |        |  +--ro yp:selection-filter-ref
       |  |        |          selection-filter-ref
       |  |        +--:(yp:within-subscription)
       |  |           +--ro (yp:filter-spec)?
       |  |              +--:(yp:datastore-subtree-filter)
```

```
     |  |              |  +--ro yp:datastore-subtree-filter?
     |  |              |          <anydata> {sn:subtree}?
     |  |              +--:(yp:datastore-xpath-filter)
     |  |                 +--ro yp:datastore-xpath-filter?
     |  |                         yang:xpath1.0 {sn:xpath}?
     |  ...
     |  +--ro (yp:update-trigger)?
     |     +--:(yp:periodic)
     |     |  +--ro yp:periodic!
     |     |     +--ro yp:period         yang:timeticks
     |     |     +--ro yp:anchor-time?   yang:date-and-time
     |     +--:(yp:on-change) {on-change}?
     |        +--ro yp:on-change!
     |           +--ro yp:dampening-period?    yang:timeticks
     |           +--ro yp:sync-on-start?       boolean
     |           +--ro yp:excluded-change*     change-type
     +---n subscription-terminated
     |  ...
     +---n subscription-suspended
        ...

  module: ietf-yang-push

    rpcs:
      +---x resync-subscription {on-change}?
         +---w input
            +---w id    sn:subscription-id

    yang-data: (for placement into rpc error responses)
      +-- resync-subscription-error
      |  +--ro reason?                   identityref
      |  +--ro period-hint?              timeticks
      |  +--ro filter-failure-hint?      string
      |  +--ro object-count-estimate?    uint32
      |  +--ro object-count-limit?       uint32
      |  +--ro kilobytes-estimate?       uint32
      |  +--ro kilobytes-limit?          uint32
      +-- establish-subscription-error-datastore
      |  +--ro reason?                   identityref
      |  +--ro period-hint?              timeticks
      |  +--ro filter-failure-hint?      string
      |  +--ro object-count-estimate?    uint32
      |  +--ro object-count-limit?       uint32
      |  +--ro kilobytes-estimate?       uint32
      |  +--ro kilobytes-limit?          uint32
      +-- modify-subscription-error-datastore
         +--ro reason?                   identityref
         +--ro period-hint?              timeticks
```

```
          +--ro filter-failure-hint?      string
          +--ro object-count-estimate?    uint32
          +--ro object-count-limit?       uint32
          +--ro kilobytes-estimate?       uint32
          +--ro kilobytes-limit?          uint32

     notifications:
       +---n push-update
       |  +--ro id?      sn:subscription-id
       |  +--ro datastore-contents?   <anydata>
       |  +--ro incomplete-update?     empty
       +---n push-change-update {on-change}?
          +--ro id?     sn:subscription-id
          +--ro datastore-changes?
          |  +--ro yang-patch
          |     +--ro patch-id        string
          |     +--ro ypatch:comment?    string
          |     +--ro ypatch:edit* [edit-id]
          |        +--ro ypatch:edit-id      string
          |        +--ro ypatch:operation    enumeration
          |        +--ro ypatch:target       target-resource-offset
          |        +--ro ypatch:point?       target-resource-offset
          |        +--ro ypatch:where?       enumeration
          |        +--ro ypatch:value?
          +--ro incomplete-update?    empty
```

Figure 6: Model structure

Selected components of the model are summarized below.

4.2. Subscription Configuration

Both configured and dynamic subscriptions are represented within the list "subscription". New parameters extending the basic subscription data model in [I-D.draft-ietf-netconf-subscribed-notifications] include:

- o The targeted datastore from which the selection is being made. The potential datastores include those from [RFC8341]. A platform may also choose to support a custom datastore.

- o A selection filter identifying yang nodes of interest within a datastore. Filter contents are specified via a reference to an existing filter, or via an in-line definition for only that subscription. Referenced filters allows an implementation to avoid evaluating filter acceptability during a dynamic subscription request. The case statement differentiates the options.

- o For periodic subscriptions, triggered updates will occur at the boundaries of a specified time interval. These boundaries can be calculated from the periodic parameters:

  * a "period" which defines the duration between push updates.

  * an "anchor-time"; update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

- o For on-change subscriptions, assuming any dampening period has completed, triggering occurs whenever a change in the subscribed information is detected. On-change subscriptions have more complex semantics that is guided by its own set of parameters:

  * a "dampening-period" specifies the interval that must pass before a successive update for the subscription is sent. If no dampening period is in effect, the update is sent immediately. If a subsequent change is detected, another update is only sent once the dampening period has passed for this subscription.

  * an "excluded-change" parameter which allows restriction of the types of changes for which updates should be sent (e.g., only add to an update record on object creation).

  * a "sync-on-start" specifies whether a complete update with all the subscribed data is to be sent at the beginning of a subscription.

4.3. YANG Notifications

4.3.1. State Change Notifications

Subscription state notifications and mechanism are reused from [I-D.draft-ietf-netconf-subscribed-notifications]. Notifications "subscription-started" and "subscription-modified" have been augmented to include the datastore specific objects.

4.3.2. Notifications for Subscribed Content

Along with the subscribed content, there are other objects which might be part of a "push-update" or "push-change-update" notification.

An "id" (that identifies the subscription) MUST be transported along with the subscribed contents. This allows a receiver to differentiate which subscription resulted in a particular update record.

A "time-of-update" which represents the time an update record snapshot was generated. A receiver MAY assume that at this point in time a publisher's objects have the values that were pushed.

An "incomplete-update" leaf. This leaf indicates that not all changes which have occurred since the last update are actually included with this update. In other words, the publisher has failed to fulfill its full subscription obligations. (For example a datastore was unable to provide the full set of datastore nodes to a publisher process.) To facilitate re-synchronization of on-change subscriptions, a publisher MAY subsequently send a "push-update" containing a full selection snapshot of subscribed data.

4.4. YANG RPCs

YANG-Push subscriptions are established, modified, and deleted using RPCs augmented from [I-D.draft-ietf-netconf-subscribed-notifications].

4.4.1. Establish-subscription RPC

The subscriber sends an establish-subscription RPC with the parameters in section 3.1. An example might look like:

```
<netconf:rpc message-id="101"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
      xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <yp:datastore xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
        xmlns:ex="http://example.com/sample-data/1.0">
      /ex:foo
    </yp:datastore-xpath-filter>
    <yp:periodic>
      <yp:period>500</yp:period>
    </yp:periodic>
  </establish-subscription>
</netconf:rpc>
```

Figure 7: Establish-subscription RPC

A positive response includes the "id" of the accepted subscription. In that case a publisher may respond:

```
<rpc-reply message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <id
      xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
       52
    </id>
</rpc-reply>
```

Figure 8: Establish-subscription positive RPC response

A subscription can be rejected for multiple reasons, including the lack of authorization to establish a subscription, no capacity to serve the subscription at the publisher, or the inability of the publisher to select datastore content at the requested cadence.

If a request is rejected because the publisher is not able to serve it, the publisher SHOULD include in the returned error hints which help a subscriber understand subscription parameters might have been accepted for the request. These hints would be included within the yang-data structure "establish-subscription-error-datastore". However even with these hints, there are no guarantee that subsequent requests will in fact be accepted.

The specific parameters to be returned as part of the RPC error response depend on the specific transport that is used to manage the subscription. For example, in the case of NETCONF [I-D.draft-ietf-netconf-netconf-event-notifications], when a subscription request is rejected, the NETCONF RPC reply would be expected to include an "rpc-error" element with the following elements:

- o "error-type" of "application".

- o "error-tag" of "operation-failed".

- o Optionally, an "error-severity" of "error".

- o Optionally, an "error-app-tag" with the value being a string that corresponds to an identity associated with the error, i.e. an identity with a base of "establish-subscription-error".

- o Optionally, "error-info" containing XML-encoded data with hints for parameter settings that might result in future RPC success per yang-data definition "establish-subscription-error-datastore".

For example, for the following request:

```
<netconf:rpc message-id="101"
  xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <establish-subscription
  xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
  xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <yp:datastore
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
    xmlns:ex="http://example.com/sample-data/1.0">
      /ex:foo
    </yp:datastore-xpath-filter>
    <yp:on-change>
      <yp:dampening-period>100</yp:dampening-period>
    </yp:on-change>
  </establish-subscription>
</netconf:rpc>
```

Figure 9: Establish-subscription request example 2

a publisher that cannot serve on-change updates but periodic updates might return the following:

```
 <rpc-reply message-id="101"
   xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
   xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
   <rpc-error>
     <error-type>application</error-type>
     <error-tag>operation-failed</error-tag>
     <error-severity>error</error-severity>
     <error-path>/yp:periodic/yp:period</error-path>
         <error-info>
       <yp:establish-subscription-error-datastore>
         <yp:reason>yp:on-change-unsupported</yp:reason>
       </yp:establish-subscription-error-datastore>
         </error-info>
   </rpc-error>
 </rpc-reply>
```

Figure 10: Establish-subscription error response example 2

4.4.2. Modify-subscription RPC

The subscriber MAY invoke the "modify-subscription" RPC for a subscription it previously established. The subscriber will include newly desired values in the "modify-subscription" RPC. Parameters not included MUST remain unmodified. Below is an example where a subscriber attempts to modify the period and datastore XPath filter of a subscription.

```
<netconf:rpc message-id="102"
   xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
   <modify-subscription
   xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
   xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>1011</id>
    <yp:datastore
    xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
      ds:operational
    </yp:datastore>
    <yp:datastore-xpath-filter
      xmlns:ex="http://example.com/sample-data/1.0">
      /ex:bar
    </yp:datastore-xpath-filter>
    <yp:periodic>
      <yp:period>250</yp:period>
    </yp:periodic>
   </modify-subscription>
</netconf:rpc>
```

Figure 11: Modify subscription request

The publisher MUST respond to the subscription modification request. If the request is rejected, the existing subscription is left unchanged, and the publisher MUST send an RPC error response. This response might have hints encapsulated within the yang-data structure "modify-subscription-error-datastore". A subscription MAY be modified multiple times.

The specific parameters to be returned in as part of the RPC error response depend on the specific transport that is used to manage the subscription. In the case of NETCONF [I-D.draft-ietf-netconf-netconf-event-notifications], when a subscription request is rejected, the NETCONF RPC reply MUST include an "rpc-error" element with the following elements:

o "error-type" of "application".

o "error-tag" of "operation-failed".

o Optionally, an "error-severity" of "error" (this MAY but does not have to be included).

o Optionally, an "error-app-tag" with the value being a string that corresponds to an identity associated with the error, i.e. an identity with a base of "modify-subscription-error".

o "error-path" pointing to the object or parameter that caused the rejection.

o Optionally, "error-info" containing XML-encoded data with hints for parameter settings that might result in future RPC success per yang-data definition "modify-subscription-error-datastore".

A configured subscription cannot be modified using "modify-subscription" RPC. Instead, the configuration needs to be edited as needed.

4.4.3. Delete-subscription RPC

To stop receiving updates from a subscription and effectively delete a subscription that had previously been established using an "establish-subscription" RPC, a subscriber can send a "delete-subscription" RPC, which takes as only input the subscription's "id". This RPC is unmodified from [I-D.draft-ietf-netconf-subscribed-notifications].

4.4.4. Resync-subscription RPC

This RPC is supported only for on-change subscriptions previously established using an "establish-subscription" RPC. For example:

```
<netconf:rpc message-id="103"
xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
  <resync-subscription
  xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
  xmlns:sn="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
    <id>1011</id>
  </resync-subscription>
</netconf:rpc>
```

Resync subscription

On receipt, a publisher must either accept the request and quickly follow with a "push-update", or send an appropriate error within an rpc error response. Within an error response, the publisher MAY include supplemental information about the reasons within the yang-data structure "resync-subscription-error".

4.4.5. YANG Module Synchronization

To make subscription requests, the subscriber needs to know the YANG datastore schemas used by the publisher, which are available via the YANG Library module, ietf-yang-library.yang from [RFC7895]. The receiver is expected to know the YANG library information before starting a subscription.

The set of modules, revisions, features, and deviations can change at run-time (if supported by the publisher implementation). For this purpose, the YANG library provides a simple "yang-library-change" notification that informs the subscriber that the library has changed. In this case, a subscription may need to be updated to take the updates into account. The receiver may also need to be informed of module changes in order to process updates regarding datastore nodes from changed modules correctly.

5. YANG Module

```
<CODE BEGINS> file "ietf-yang-push@2018-10-22.yang"
module ietf-yang-push {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yang-push";
  prefix yp;

  import ietf-yang-types {
```

```
     prefix yang;
         reference
           "RFC 6991: Common YANG Data Types";
   }
   import ietf-subscribed-notifications {
     prefix sn;
     reference
       "draft-ietf-netconf-subscribed-notifications:
        Customized Subscriptions to a Publisher's Event Streams

        NOTE TO RFC Editor: Please replace above reference to
        draft-ietf-netconf-subscribed-notifications with RFC number
        when published (i.e. RFC xxxx).";
   }
   import ietf-datastores {
     prefix ds;
     reference
           "RFC 8342: Network Management Datastore Architecture (NMDA)";
   }
   import ietf-restconf   {
     prefix rc;
     reference
       "RFC 8040: RESTCONF Protocol";
   }

   import ietf-yang-patch {
     prefix ypatch;
     reference
       "RFC 8072: YANG Patch";
   }
   organization "IETF";
   contact
     "WG Web:   <http://tools.ietf.org/wg/netconf/>
      WG List:  <mailto:netconf@ietf.org>

      Editor:   Alexander Clemm
                <mailto:ludwig@clemm.org>

      Editor:   Eric Voit
                <mailto:evoit@cisco.com>

      Editor:   Alberto Gonzalez Prieto
                <mailto:agonzalezpri@vmware.com>

      Editor:   Ambika Prasad Tripathy
                <mailto:ambtripa@cisco.com>

      Editor:   Einar Nilsen-Nygaard
```

```
                <mailto:einarnn@cisco.com>

      Editor:   Andy Bierman
                <mailto:andy@yumaworks.com>

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

   description
     "This module contains YANG specifications for YANG push.

     Copyright (c) 2018 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

     This version of this YANG module is part of
     draft-ietf-netconf-yang-push-20; see the RFC itself for full
     legal notices.

     NOTE TO RFC EDITOR: Please replace above reference to
     draft-ietf-netconf-yang-push-20 with RFC number when published
     (i.e. RFC xxxx).";

   revision 2018-10-22 {
     description
       "Initial revision.
       NOTE TO RFC EDITOR:
       (1)Please replace the above revision date to
       the date of RFC publication when published.
       (2) Please replace the date in the file name
       (ietf-yang-push@2018-10-22.yang) to the date of RFC
       publication.
       (3) Please replace the following reference to
       draft-ietf-netconf-yang-push-20 with RFC number when
       published (i.e. RFC xxxx).";
     reference
       "draft-ietf-netconf-yang-push-20";
   }

  /*
   * FEATURES
   */
```

```
   feature on-change {
     description
       "This feature indicates that on-change triggered
       subscriptions are supported.";
   }

  /*
   * IDENTITIES
   */

   /* Error type identities for datastore subscription  */

   identity resync-subscription-error {
      description
       "Problem found while attempting to fulfill an
       'resync-subscription' RPC request. ";
   }

   identity cant-exclude {
     base sn:establish-subscription-error;
     description
       "Unable to remove the set of 'excluded-changes'.  This means
       the publisher is unable to restrict 'push-change-update's to
       just the change types requested for this subscription.";
   }

   identity datastore-not-subscribable {
     base sn:establish-subscription-error;
     base sn:subscription-terminated-reason;
     description
       "This is not a subscribable datastore.";
   }

   identity no-such-subscription-resync {
     base resync-subscription-error;
     description
       "Referenced subscription doesn't exist. This may be as a
       result of a non-existent subscription ID, an ID which
       belongs to another subscriber, or an ID for configured
       subscription.";
   }

   identity on-change-unsupported {
     base sn:establish-subscription-error;
     description
       "On-change is not supported for any objects which are
       selectable by this filter.";
   }
```

```
   identity on-change-sync-unsupported {
     base sn:establish-subscription-error;
     description
       "Neither sync on start nor resynchronization are supported
       for this subscription.  This error will be used for two
       reasons. First if an 'establish-subscription' RPC includes
       'sync-on-start', yet the publisher can't support sending a
       'push-update' for this subscription for reasons other than
       'on-change-unsupported' or 'sync-too-big'.  And second,
       if the 'resync-subscription' RPC is invoked either for an
       existing periodic subscription, or for an on-change
       subscription which can't support resynchronization.";
   }

   identity period-unsupported {
     base sn:establish-subscription-error;
     base sn:modify-subscription-error;
     base sn:subscription-suspended-reason;
     description
       "Requested time period or dampening-period is too short. This
       can be for both periodic and on-change subscriptions (with or
       without dampening.) Hints suggesting alternative periods may
       be returned as supplemental information.";
   }

   identity update-too-big {
     base sn:establish-subscription-error;
     base sn:modify-subscription-error;
     base sn:subscription-suspended-reason;
     description
       "Periodic or on-change push update datatrees exceed a maximum
       size limit.  Hints on estimated size of what was too big may
       be returned as supplemental information.";
   }

   identity sync-too-big {
     base sn:establish-subscription-error;
     base sn:modify-subscription-error;
     base resync-subscription-error;
     base sn:subscription-suspended-reason;
     description
       "Sync-on-start or resynchronization datatree exceeds a
       maximum size limit.  Hints on estimated size of what was too
       big may be returned as supplemental information.";
   }

   identity unchanging-selection {
     base sn:establish-subscription-error;
```

```
     base sn:modify-subscription-error;
     base sn:subscription-terminated-reason;
     description
       "Selection filter is unlikely to ever select datatree nodes.
       This means that based on the subscriber's current access
       rights, the publisher recognizes that the selection filter is
       unlikely to ever select datatree nodes which change. Examples
       for this might be that node or subtree doesn't exist, read
       access is not permitted for a receiver, or static objects
       that only change at reboot have been chosen.";
   }

   /*
    * TYPE DEFINITIONS
    */

   typedef change-type {
     type enumeration {
       enum "create" {
         description
           "A change that refers to the creation of a new datastore
           node.";
       }
       enum "delete" {
         description
           "A change that refers to the deletion of a datastore
           node.";
       }
       enum "insert" {
         description
           "A change that refers to the insertion of a new
           user-ordered datastore node.";
       }
       enum "move" {
         description
           "A change that refers to a reordering of the target
           datastore node";
       }
       enum "replace" {
         description
           "A change that refers to a replacement of the target
           datastore node's value.";
       }
     }
     description
       "Specifies different types of datastore changes.";
     reference
       "RFC 8072 section 2.5, with a delta that it is valid for a
```

```
        receiver to process an update record which performs a create
        operation on a datastore node the receiver believes exists,
        or to process a delete on a datastore node the receiver
        believes is missing.";
   }

   typedef selection-filter-ref {
     type leafref {
       path "/sn:filters/yp:selection-filter/yp:filter-id";
     }
     description
       "This type is used to reference a selection filter.";
   }

   /*
    * GROUP DEFINITIONS
    */

   grouping datastore-criteria {
     description
       "A grouping to define criteria for which selected objects
       from  a targeted datastore should be included in push
       updates.";
     leaf datastore {
        type identityref {
          base ds:datastore;
        }
        mandatory true;
        description
          "Datastore from which to retrieve data.";
     }
     uses selection-filter-objects;
   }

   grouping selection-filter-types {
     description
       "This grouping defines the types of selectors for objects
       from a datastore.";
     choice filter-spec {
       description
         "The content filter specification for this request.";
       anydata datastore-subtree-filter {
         if-feature "sn:subtree";
         description
           "This parameter identifies the portions of the
           target datastore to retrieve.";
         reference
           "RFC 6241: Network Configuration Protocol, Section 6.";
```

```
        }
        leaf datastore-xpath-filter {
          if-feature "sn:xpath";
          type yang:xpath1.0;
          description
            "This parameter contains an XPath expression identifying
            the portions of the target datastore to retrieve.

            If the expression returns a node-set, all nodes in the
            node-set are selected by the filter.  Otherwise, if the
            expression does not return a node-set, the filter
            doesn't select any nodes.

            The expression is evaluated in the following XPath
            context:

             o  The set of namespace declarations are those in scope
                on the 'datastore-xpath-filter' leaf element.

             o  The set of variable bindings is empty.

             o  The function library is the core function library, and
                the XPath functions defined in section 10 in RFC 7950.

             o  The context node is the root node of the target
                datastore.";
        }
      }
    }

    grouping selection-filter-objects {
      description
        "This grouping defines a selector for objects from a
        datastore.";
      choice selection-filter {
        description
          "The source of the selection filter applied to the
          subscription. This will come either referenced from a
          global list, or be provided within the subscription
          itself.";
        case by-reference {
          description
            "Incorporate a filter that has been configured
            separately.";
          leaf selection-filter-ref {
            type selection-filter-ref;
            mandatory true;
            description
```

```
            "References an existing selection filter which is to be
            applied to the subscription.";
          }
        }
        case within-subscription {
          description
            "Local definition allows a filter to have the same
            lifecycle as the subscription.";
          uses selection-filter-types;
        }
      }
    }

    grouping update-policy-modifiable {
      description
        "This grouping describes the datastore specific subscription
        conditions that can be changed during the lifetime of the
        subscription.";
      choice update-trigger {
        when "../sn:target/yp:datastore";
            mandatory true;
        description
          "Defines necessary conditions for sending an event record to
          the subscriber.";
        case periodic {
          container periodic {
            presence "indicates a periodic subscription";
            description
              "The publisher is requested to notify periodically the
              current values of the datastore as defined by the
              selection filter.";
            leaf period {
              type yang:timeticks;
              mandatory true;
              description
                "Duration of time which should occur between periodic
                push updates, in one hundredths of a second.";
            }
            leaf anchor-time {
              type yang:date-and-time;
              description
                "Designates a timestamp before or after which a
                series of periodic push updates are determined. The
                next update will take place at a whole multiple
                interval from the anchor time.  For example, for an
                anchor time is set for the top of a particular
                minute and a period interval of a minute, updates
                will be sent at the top of every minute this
```

```
               subscription is active.";
           }
         }
       }
       case on-change {
         if-feature "on-change";
         container on-change {
           presence "indicates an on-change subscription";
           description
             "The publisher is requested to notify changes in
             values in the datastore subset as defined by a
             selection filter.";
           leaf dampening-period {
             type yang:timeticks;
             default 0;
             description
               "Specifies the minimum interval between the assembly
               of successive update records for a single receiver
               of a subscription. Whenever subscribed objects
               change, and a dampening period interval (which may
               be zero) has elapsed since the previous update
               record creation for a receiver, then any subscribed
               objects and properties which have changed since the
               previous update record will have their current
               values marshalled and placed into a new update
              6 record.";
           }
         }
       }
     }
   }

   grouping update-policy {
     description
       "This grouping describes the datastore specific subscription
        conditions of a subscription.";
     uses update-policy-modifiable {
       augment "update-trigger/on-change/on-change" {
         description
           "Includes objects not modifiable once subscription is
            established.";
         leaf sync-on-start {
           type boolean;
           default "true";
           description
             "When this object is set to false, it restricts an
             on-change subscription from sending push-update
             notifications.  When false, pushing a full selection
```

```
            per the terms of the selection filter MUST NOT be done
            for this subscription. Only updates about changes,
            i.e. only push-change-update notifications are sent.
            When true (default behavior), in order to facilitate a
            receiver's synchronization, a full update is sent when
            the subscription starts using a push-update
            notification.  After that, push-change-update
            notifications are exclusively sent unless the
            publisher chooses to resync the subscription via a new
            push-update notification.";
        }
        leaf-list excluded-change {
          type change-type;
          description
            "Use to restrict which changes trigger an update.
            For example, if modify is excluded, only creation and
            deletion of objects is reported.";
        }
      }
    }
  }

  grouping hints {
    description
      "Parameters associated with some error for a subscription
      made upon a datastore.";
    leaf period-hint {
      type yang:timeticks;
      description
        "Returned when the requested time period is too short. This
        hint can assert a viable period for either a periodic push
        cadence or an on-change dampening interval.";
    }
    leaf filter-failure-hint {
      type string;
        description
          "Information describing where and/or why a provided filter
          was unsupportable for a subscription.";
    }
    leaf object-count-estimate {
      type uint32;
      description
        "If there are too many objects which could potentially be
        returned by the selection filter, this identifies the
        estimate of the number of objects which the filter would
        potentially pass.";
    }
    leaf object-count-limit {
```

```
        type uint32;
        description
          "If there are too many objects which could be returned by
          the selection filter, this identifies the upper limit of
          the publisher's ability to service for this subscription.";
      }
      leaf kilobytes-estimate {
        type uint32;
        description
          "If the returned information could be beyond the capacity
          of the publisher, this would identify the data size which
          could result from this selection filter.";
      }
      leaf kilobytes-limit {
        type uint32;
        description
          "If the returned information would be beyond the capacity
          of the publisher, this identifies the upper limit of the
          publisher's ability to service for this subscription.";
      }
    }


    /*
     * RPCs
     */

     rpc resync-subscription {
      if-feature "on-change";
      description
        "This RPC allows a subscriber of an active on-change
        subscription to request a full push of objects.
        A successful invocation results in a push-update of all
        datastore nodes that the subscriber is permitted to access.
        This RPC can only be invoked on the same session on which the
        subscription is currently active.  In case of an error, a
        resync-subscription-error is sent as part of an error
        response.";
      input {
        leaf id {
          type sn:subscription-id;
          mandatory true;
          description
            "Identifier of the subscription that is to be resynced.";
        }
      }
    }
```

```
   rc:yang-data resync-subscription-error {
     container resync-subscription-error {
       description
         "If a 'resync-subscription' RPC fails, the subscription is
         not resynced and the RPC error response MUST indicate the
         reason for this failure. This yang-data MAY be inserted as
         structured data within a subscription's RPC error response
         to indicate the failure reason.";
       leaf reason {
         type identityref {
           base resync-subscription-error;
         }
         mandatory true;
         description
           "Indicates the reason why the publisher has declined a
           request for subscription resynchronization.";
       }
       uses hints;
     }
   }

   augment "/sn:establish-subscription/sn:input" {
     when "sn:target/yp:datastore";
         description
       "This augmentation adds additional subscription parameters
       that apply specifically to datastore updates to RPC input.";
     uses update-policy;
   }

   augment "/sn:establish-subscription/sn:input/sn:target" {
     description
       "This augmentation adds the datastore as a valid target
       for the subscription to RPC input.";
     case datastore {
       description
         "Information specifying the parameters of an request for a
         datastore subscription.";
       uses datastore-criteria;
     }
   }

   rc:yang-data establish-subscription-datastore-error-info {
     container establish-subscription-datastore-error-info {
       description
         "If any 'establish-subscription' RPC parameters are
         unsupportable against the datastore, a subscription is not
         created and the RPC error response MUST indicate the reason
         why the subscription failed to be created. This yang-data
```

```
         MAY be inserted as structured data within a subscription's
         RPC error response to indicate the failure reason.  This
         yang-data MUST be inserted if hints are to be provided back
         to the subscriber.";
       leaf reason {
         type identityref {
           base sn:establish-subscription-error;
         }
         description
           "Indicates the reason why the subscription has failed to
           be created to a targeted datastore.";
       }
       uses hints;
     }
   }

   augment "/sn:modify-subscription/sn:input" {
     when "sn:target/yp:datastore";
     description
       "This augmentation adds additional subscription parameters
        specific to datastore updates.";
     uses update-policy-modifiable;
   }

   augment "/sn:modify-subscription/sn:input/sn:target" {
     description
       "This augmentation adds the datastore as a valid target
       for the subscription to RPC input.";
     case datastore {
       description
         "Information specifying the parameters of an request for a
          datastore subscription.";
       uses selection-filter-objects;
     }
   }

   rc:yang-data modify-subscription-datastore-error-info {
     container modify-subscription-datastore-error-info {
       description
         "This yang-data MAY be provided as part of a subscription's
         RPC error response when there is a failure of a
         'modify-subscription' RPC which has been made against a
         datastore.  This yang-data MUST be used if hints are to be
         provides back to the subscriber.";
       leaf reason {
         type identityref {
           base sn:modify-subscription-error;
         }
```

```
         description
           "Indicates the reason why the subscription has failed to
           be modified.";
       }
       uses hints;
     }
   }

   /*
    * NOTIFICATIONS
    */

   notification push-update {
     description
       "This notification contains a push update, containing data
       subscribed to via a subscription. This notification is sent
       for periodic updates, for a periodic subscription.  It can
       also be used for synchronization updates of an on-change
       subscription. This notification shall only be sent to
       receivers of a subscription.  It does not constitute a
       general-purpose notification that would be subscribable as
       part of the NETCONF event stream by any receiver.";
     leaf id {
       type sn:subscription-id;
       description
         "This references the subscription which drove the
         notification to be sent.";
     }
     anydata datastore-contents {
       description
         "This contains the updated data.  It constitutes a snapshot
         at the time-of-update of the set of data that has been
         subscribed to.  The snapshot corresponds to the same
         snapshot that would be returned in a corresponding get
         operation with the same selection filter parameters
         applied.";
     }
     leaf incomplete-update {
       type empty;
       description
         "This is a flag which indicates that not all datastore
         nodes subscribed to are included with this update. In
         other words, the publisher has failed to fulfill its full
         subscription obligations, and despite its best efforts is
         providing an incomplete set of objects.";
     }
   }
```

```
   notification push-change-update {
     if-feature "on-change";
     description
       "This notification contains an on-change push update. This
       notification shall only be sent to the receivers of a
       subscription; it does not constitute a general-purpose
       notification.";
     leaf id {
       type sn:subscription-id;
       description
         "This references the subscription which drove the
         notification to be sent.";
     }
     container datastore-changes {
       description
         "This contains the set of datastore changes of the
         target datastore starting at the time of the
         previous update, per the terms of the subscription.
         The datastore changes are encoded per RFC 8027
         (YANG Patch).";
           uses ypatch:yang-patch;
     }
     leaf incomplete-update {
       type empty;
       description
         "The presence of this object indicates not all changes which
         have occurred since the last update are included with this
         update.  In other words, the publisher has failed to
         fulfill its full subscription obligations, for example in
         cases where it was not able to keep up with a change
         burst.";
     }
   }

   augment "/sn:subscription-started" {
     description
       "This augmentation adds datastore-specific objects to
        the notification that a subscription has started.";
     uses update-policy;
   }

   augment "/sn:subscription-started/sn:target" {
     description
       "This augmentation allows the datastore to be included as
       part of the notification that a subscription has started.";
     case datastore {
        uses datastore-criteria {
           refine "selection-filter/within-subscription" {
```

```
           description
             "Specifies the selection filter and where it
             originated from. If the 'selection-filter-ref' is
             populated, the filter within the subscription came
             from the 'filters' container.  Otherwise it is
             populated in-line as part of the subscription itself.";
         }
       }
     }
   }

   augment "/sn:subscription-modified" {
     description
       "This augmentation adds datastore-specific objects to
       the notification that a subscription has been modified.";
     uses update-policy;
   }

   augment "/sn:subscription-modified/sn:target" {
     description
       "This augmentation allows the datastore to be included as
       part of the notification that a subscription has been
       modified.";
     case datastore {
        uses datastore-criteria {
           refine "selection-filter/within-subscription" {
           description
             "Specifies where the selection filter, and where it
             came from within the subscription and then populated
             within this notification. If the
             'selection-filter-ref' is populated, the filter within
             the subscription came from the 'filters' container.
             Otherwise it is populated in-line as part of the
             subscription itself.";
         }
       }
     }
   }

   /*
    * DATA NODES
    */

   augment "/sn:filters" {
     description
       "This augmentation allows the datastore to be included as part
       of the selection filtering criteria for a subscription.";
     list selection-filter {
```

```
       key "filter-id";
       description
         "A list of pre-configured filters that can be applied
         to datastore subscriptions.";
       leaf filter-id {
         type string;
         description
           "An identifier to differentiate between selection
           filters.";
       }
       uses selection-filter-types;
     }
   }

   augment "/sn:subscriptions/sn:subscription" {
     when "sn:target/yp:datastore";
     description
       "This augmentation adds many datastore specific objects to a
       subscription.";
     uses update-policy;
   }
   augment "/sn:subscriptions/sn:subscription/sn:target" {
     description
       "This augmentation allows the datastore to be included as
       part of the selection filtering criteria for a subscription.";
     case datastore {
        uses datastore-criteria;
     }
   }
 }

 <CODE ENDS>
```

6. IANA Considerations

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

URI: urn:ietf:params:xml:ns:yang:ietf-yang-push
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 [RFC6020]:

Name: ietf-yang-push
Namespace: urn:ietf:params:xml:ns:yang:ietf-yang-push

Prefix: yp
Reference: draft-ietf-netconf-yang-push-20.txt (RFC form)

7. 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 Network Configuration 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. (It should be noted that the YANG module augments the YANG module from [I-D.draft-ietf-netconf-subscribed-notifications]. All security considerations that are listed there are relevant also for datastore subscriptions. In the following, we focus on the data nodes that are newly introduced here.)

- Subtree "selection-filter" under container "filters": This subtree allows to specify which objects or subtrees to include in a datastore subscription. An attacker could attempt to modify the filter. For example, the filter might be modified to result in very few objects being filtered in order to attempt to overwhelm the receiver. Alternatively, the filter might be modified to result in certain objects to be excluded from updates, in order to have certain changes go unnoticed.

- Subtree "datastore" in choice "target" in list "subscription": Analogous to "selection filter", an attacker might attempt to modify the objects being filtered in order to overwhelm a receiver with a larger volume of object updates than expected, or to have certain changes go unnoticed.

- Choice "update-trigger" in list "subscription": By modifying the update trigger, an attacker might alter the updates that are being sent in order to confuse a receiver, to withhold certain updates to be sent to the receiver, and/or to overwhelm a receiver. For example, an attacker might modify the period with which updates are reported for a periodic subscription, or it might modify the dampening period for an on-change subscription, resulting in greater delay of successive updates (potentially affecting responsiveness of applications that depend on the updates) or in a high volume of updates (to exhaust receiver resources).

- RPC "resync-subscription": This RPC allows a subscriber of an on-change subscription to request a full push of objects in the subscription's scope. This can result in a large volume of data. An attacker could attempt to use this RPC to exhaust resources on the server to generate the data, and attempt to overwhelm a receiver with the resulting data volume.

8. Acknowledgments

For their valuable comments, discussions, and feedback, we wish to acknowledge Tim Jenkins, Martin Bjorklund, Kent Watsen, Susan Hares, Yang Geng, Peipei Guo, Michael Scharf, Guangying Zheng, Tom Petch, Henk Birkholz, Reshad Rahman, Qin Wu, Rohit Ranade, and Rob Wilton.

9. References

9.1. Normative References

[I-D.draft-ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Gonzalez Prieto, A., Tripathy, A., and E. Nilsen-Nygaard, "Custom Subscription to Event Streams", draft-ietf-netconf-subscribed-notifications-13 (work in progress), August 2018.

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

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

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <https://www.rfc-editor.org/info/rfc6470>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8072] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", RFC 8072, DOI 10.17487/RFC8072, February 2017, <https://www.rfc-editor.org/info/rfc8072>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

9.2. Informative References

[I-D.draft-ietf-netconf-netconf-event-notifications] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "NETCONF Support for Event Notifications", August 2018.

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

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <https://www.rfc-editor.org/info/rfc7923>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

Appendix A. Appendix A: Subscription Errors

A.1. RPC Failures

Rejection of an RPC for any reason is indicated by via RPC error response from the publisher. Valid RPC errors returned include both existing transport layer RPC error codes, such as those seen with NETCONF in [RFC6241], as well as subscription specific errors such as those defined within the YANG model. As a result, how subscription errors are encoded within an RPC error response is transport dependent.

References to specific identities within the either the subscribed-notifications YANG model or the yang-push YANG model may be returned as part of the error responses resulting from failed attempts at datastore subscription. Following are valid errors per RPC (note: throughout this section the prefix 'sn' indicates an item imported from the subscribed-notifications.yang model):

```
establish-subscription         modify-subscription
----------------------         -------------------
 cant-exclude                   sn:filter-unsupported
 datastore-not-subscribable     sn:insufficient-resources
 sn:dscp-unavailable            sn:no-such-subscription
 sn:filter-unsupported          period-unsupported
 sn:insufficient-resources      update-too-big
 on-change-unsupported          sync-too-big
 on-change-sync-unsupported     unchanging-selection
 period-unsupported
 update-too-big                resync-subscription
 sync-too-big                  --------------------
 unchanging-selection           no-such-subscription-resync
                                sync-too-big

delete-subscription            kill-subscription
----------------------         -----------------
 sn:no-such-subscription        sn:no-such-subscription
```

There is one final set of transport independent RPC error elements included in the YANG model. These are the following four yang-data structures for failed datastore subscriptions:

1. yang-data establish-subscription-error-datastore
   This MUST be returned if information identifying the reason for an RPC error has not been placed elsewhere within the transport portion of a failed "establish-subscription" RPC response. This MUST be sent if hints are included.

2. yang-data modify-subscription-error-datastore
   This MUST be returned if information identifying the reason for an RPC error has not been placed elsewhere within the transport portion of a failed "modifiy-subscription" RPC response. This MUST be sent if hints are included.

3. yang-data sn:delete-subscription-error
   This MUST be returned if information identifying the reason for an RPC error has not been placed elsewhere within the transport portion of a failed "delete-subscription" or "kill-subscription" RPC response.

4. yang-data resync-subscription-error
   This MUST be returned if information identifying the reason for an RPC error has not been placed elsewhere within the transport portion of a failed "resync-subscription" RPC response.

A.2. Notifications of Failure

A subscription may be unexpectedly terminated or suspended independent of any RPC or configuration operation. In such cases, indications of such a failure MUST be provided. To accomplish this, the following types of error identities may be returned within the corresponding subscription state change notification:

```
subscription-terminated        subscription-suspended
-----------------------        ----------------------
 datastore-not-subscribable     sn:insufficient-resources
 sn:filter-unavailable          period-unsupported
 sn:no-such-subscription        update-too-big
 sn:suspension-timeout          synchronization-size
 unchanging-selection
```

Appendix B. Changes Between Revisions

(To be removed by RFC editor prior to publication)

v19 - v20

- o Minor updates per WGLC comments.

v18 - v19

- o Minor updates per WGLC comments.

v17 - v18

- o Minor updates per WGLC comments.

v16 - v17

- o Minor updates to YANG module, incorporating comments from Tom Petch.
- o Updated references.

v15 - v16

- o Updated security considerations.
- o Updated references.
- o Addressed comments from last call review, specifically comments received from Martin Bjorklund.

v14 - v15

- Minor text fixes. Includes a fix to on-change update calculation to cover churn when an object changes to and from a value during a dampening period.

v13 - v14

- Minor text fixes.

v12 - v13

- Hint negotiation models now show error examples.

- yang-data structures for rpc errors.

v11 - v12

- Included Martin's review clarifications.

- QoS moved to subscribed-notifications

- time-of-update removed as it is redundant with RFC5277's eventTime, and other times from notification-messages.

- Error model moved to match existing implementations

- On-change notifiable removed, how to do this is implementation specific.

- NMDA model supported. Non NMDA version at https://github.com/netconf-wg/yang-push/

v10 - v11

- Promise model reference added.

- Error added for no-such-datastore

- Inherited changes from subscribed notifications (such as optional feature definitions).

- scrubbed the examples for proper encodings

v09 - v10

- Returned to the explicit filter subtyping of v00-v05 o identityref to ds:datastore made explicit

o Returned ability to modify a selection filter via RPC.

v08 - v09

o Minor tweaks cleaning up text, removing appendicies, and making reference to revised-datastores.

o Subscription-id (now:id) optional in push updates, except when encoded in RFC5277, Section 4 one-way notification.

o Finished adding the text descibing the resync subscription RPC.

o Removed relationships to other drafts and future technology appendicies as this work is being explored elsewhere.

o Deferred the multi-line card issue to new drafts

o Simplified the NACM interactions.

v07 - v08

o Updated YANG models with minor tweaks to accommodate changes of ietf-subscribed-notifications.

v06 - v07

o Clarifying text tweaks.

o Clarification that filters act as selectors for subscribed datastore nodes; support for value filters not included but possible as a future extension

o Filters don't have to be matched to existing YANG objects

v05 - v06

o Security considerations updated.

o Base YANG model in [subscribe] updated as part of move to identities, YANG augmentations in this doc matched up

o Terms refined and text updates throughout

o Appendix talking about relationship to other drafts added.

o Datastore replaces stream

o Definitions of filters improved

v04 to v05

o Referenced based subscription document changed to Subscribed Notifications from 5277bis.

o Getting operational data from filters

o Extension notifiable-on-change added

o New appendix on potential futures. Moved text into there from several drafts.

o Subscription configuration section now just includes changed parameters from Subscribed Notifications

o Subscription monitoring moved into Subscribed Notifications

o New error and hint mechanisms included in text and in the yang model.

o Updated examples based on the error definitions

o Groupings updated for consistency

o Text updates throughout

v03 to v04

o Updates-not-sent flag added

o Not notifiable extension added

o Dampening period is for whole subscription, not single objects

o Moved start/stop into rfc5277bis

o Client and Server changed to subscriber, publisher, and receiver

o Anchor time for periodic

o Message format for synchronization (i.e. sync-on-start)

o Material moved into 5277bis

o QoS parameters supported, by not allowed to be modified by RPC o Text updates throughout

Authors' Addresses

Alexander Clemm
Huawei

Email: ludwig@clemm.org

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Alberto Gonzalez Prieto
VMware

Email: agonzalezpri@vmware.com

Ambika Prasad Tripathy
Cisco Systems

Email: ambtripa@cisco.com

Einar Nilsen-Nygaard
Cisco Systems

Email: einarnn@cisco.com

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Balazs Lengyel
Ericsson

Email: balazs.lengyel@ericsson.com

NETCONF Working Group   W. Zheng
Internet-Draft   Q. Wu
Intended status: Standards Track   Huawei
Expires: January 2, 2019   July 1, 2018

Inline Action Capability for NETCONF
draft-zheng-netconf-inline-action-capability-01

Abstract

This document defines capability based extension to NETCONF protocol that enables modification of <edit-config> operation and <edit-data> operation to accept action parameters and attributes and allows multiple sub-operations with inline action operation that apply to either different or the same conceptual node in the underlying data model in one transaction.

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. Inline-Action Capability 3
2.1. Description 3
2.2. Dependencies 3
2.3. Capability Identifier 3
2.4. New Operations 3
2.5. Modifications to Existing Operations 3
2.5.1. <edit-config> and <edit-data> 3
3. Security Considerations 5
4. IANA Considerations 6
4.1. NETCONF Capability URN 6
5. Normative References 6
Authors' Addresses 7

1. Introduction

YANG 1.1 define the syntax and semantics of version 1.1 of the YANG language, which can be used to model configuration data, state data, Remote Procedure Calls, and notifications for network management protocols. One key Difference from YANG 1.0, is a new statement "action", is added to YANG 1.1 to define operations connected to a specific container or list data node in a datastore. However which data node is applied to which configuration datastore is not specified under "action".

The <edit-data> operation defined in [I-D.ietf-netconf-nmda-netconf] and the <edit-config> operation defined in [RFC6241], are used to changes the contents of a writable Datastore. Containers and List entries can be created, deleted, replaced, and modified through <edit-config> by using the "operation" attribute in the container's and List's XML element. However the action is not part of <config> element in either <edit- data> operation or <edit-config> operation. Therefore the action operation and <edit-data> operation or <edit-config> operation connected to the same data node can not automatically handled in sequence in one transaction.

This document defines capability based extension to NETCONF protocol that enables modification of <edit-config> operation and <edit-data> operation to accept action parameters and allows multiple sub-operations with inline action operation that apply to different or same conceptual node in the underlying data model in one transaction.

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. Inline-Action Capability

2.1. Description

The :inline-action capability indicates that the device supports Inline-action operation within <edit-config> and <edit-data> operation on writable configuration datastore. In other words, the device supports <inline-action> operation is included in <edit-config> and <edit-data> operations.

2.2. Dependencies

None.

2.3. Capability Identifier

The :inline-action capability is identified by the following capability string:

urn:ietf:params:netconf:capability:inline-action:1.1

2.4. New Operations

None.

2.5. Modifications to Existing Operations

2.5.1. <edit-config> and <edit-data>

The :inline-action:1.1 capability modifies the <edit-config> <edit-data>operation to accept the <action> parameter and <action> attribute value within operation attribute.

As described in [RFC6241], "operation" attribute is defined in a element within <config> subtree and identify the point in the configuration to perform the operation and MAY appear on multiple elements throughout the <config> subtree. In this document, a new "operation" attribute value is added as follows:

inline-action: The configuration data identified by the element containing this attribute is accompanied with action operation applied to a subset of configuration within <config> subtree before edit operation is applied to the same configuration at the corresponding level in the configuration datastore identified by the <target> parameter.

In addition,the inline-action operation attribute and other "operation" attributes can apply to the same conceptual nodes in the underlying data model. The assumption is the inline-action operation attribute and other "operation" attributes applied to the same conceptual nodes will not cause unexpected operation results.

As described in [RFC6241], the config subtree is expressed as A hierarchy of configuration data as defined by one of the device's data models. The contents MUST follow the constraints of that data model, as defined by its capability definition. If inline action capability is supported, the config subtree may contain a schema node with the name "input" and a schema node with the name "output" connected to a specific container or list data node containing action element in a datastore.

Example:

```
     container interfaces {
     list interface {
       key "name";
       config true;

       leaf name {
         type string;
       }

       leaf mtu {
         type uint32;
       }
     }
     action ifstatenable {
         input {
           leaf enable {
             type boolean;
             mandatory true;
            }
          }
     }
}
```

Enable ifstatistics on 1000 interfaces from the running configuration before setting the MTU to 1500 on an interface named "Ethernet0/0" and 1000 on an interface named "Ethernet0/1" in the running configuration:

```
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
     <edit-config>
       <target>
         <running/>
       </target>
       <default-operation>none</default-operation>
       <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.1">
         <top xmlns="http://example.com/schema/1.2/config">
          <interfaces>
           <interface xc:operation="merge">
             <name>Ethernet0/0</name>
             <mtu>1500</mtu>
           </interface>
           <interface>
             <name>Ethernet0/1</name>
             <mtu>1000</mtu>
           </interface>
           <action xmlns="http://example.com/schema/1.2/config">
           <ifstatenable xc:operation="action">
              <input>
              <enable>true</enable>
              </input>
           </ifstatenable>
           </action>
         </interfaces>
         </top>
       </config>
     </edit-config>
   </rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.1">
  <ok/>
</rpc-reply>
```

3. Security Considerations

This document does not introduce any security vulnerability besides on defined in [RFC6241].

4. IANA Considerations

4.1. NETCONF Capability URN

IANA has created and now maintains a registry "Network Configuration Protocol (NETCONF) Capability URNs" that allocates NETCONF capability identifiers. Additions to the registry require IETF Standards Action.

IANA has added the following capabilities to the registry:

```
Index
   Capability Identifier
------------------------
:inline-action:1.1
   urn:ietf:params:netconf:capability:inline-action:1.1
```

5. Normative References

[I-D.ietf-netconf-nmda-netconf]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", draft-ietf-netconf-nmda-netconf-06 (work in progress), May 2018.

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

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

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <https://www.rfc-editor.org/info/rfc6021>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <https://www.rfc-editor.org/info/rfc6470>.

Authors' Addresses

Walker Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhengguangying@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

NETCONF
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2019

T. Zhou
G. Zheng
Huawei
E. Voit
Cisco Systems
A. Clemm
Huawei
A. Bierman
YumaWorks

October 18, 2018

Subscription to Multiple Stream Originators

draft-zhou-netconf-multi-stream-originators-03

Abstract

This document describes the distributed data collection mechanism that allows multiple data streams to be managed using a single subscription. Specifically, multiple data streams are pushed directly to the collector without passing through a broker for internal consolidation.

Requirements Language

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

Status of This Memo

Table of Contents

1. Introduction 2
2. Use Cases 3
2.1. Use Case 1: Data Collection from Devices with Main-board and Line-cards 3
2.2. Use Case 2: IoT Data Collection 4
3. Terminologies 5
4. Solution Overview 6
5. Subscription Decomposition 8
6. Publication Composition 9
7. Subscription State Change Notifications 10
8. IANA Considerations 10
9. Security Considerations 10
10. Acknowledgements 10
11. References 10
11.1. Normative References 11
11.2. Informative References 11
Appendix A. Change Log 12
Authors' Addresses 12

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.

```
                  +-------------------------------------+
                  |              collector              |
                  +------^-----------^-----------^------+
                         |           |           |
                         |           |           |
                  +-------------------------------------+
                  |      |           |           |      |
                  |      |     +-----+------+    |      |
                  |      |     | main board |    |      |
                  |      |     +--^-----^---+    |      |
                  |      |        |     |        |      |
                  |      |    +---+     +---+    |      |
                  |      |    |             |    |      |
                  | +----+----+---+     +---+----+----+ |
                  | | line card 1 |     | line card 2 | |
                  | +-------------+     +-------------+ |
                  |               device                |
                  +-------------------------------------+
```

Fig. 1 Data Collection from Devices with Main-board and Line-cards

2.2. Use Case 2: IoT Data Collection

In the IoT data collection scenario, as shown in the following figure, collector usually cannot access to IoT nodes directly, but is isolated by the border router. So the collector subscribes data from the border router, and let the border router to disassemble the subscription to corresponding IoT nodes. The border router is typically the traffic convergence point. It’s intuitive to treat the border router as a broker assembling the data collected from the IoT nodes and forwarding to the collector[I-D.ietf-core-coap-pubsub]. However, the border router is not so powerful on data assembling as a network device. It’s more efficient for the collector, which may be a server or even a cluster, to assemble the subscribed data if possible. In this case, push servers that reside in IoT nodes can stream data to the collector directly while traffic only passes through the border router.

```
+--------------------------------+
|           collector            |
+---^-----------^------------^---+
    |           |            |
    |           |            |
    |           |            |
    |   +-------+--------+   |
    |   |  border router |   |
    |   +----^------^----+   |
    |        |      |        |
    |        |      |        |
    |    +---+      +---+    |
    |    |              |    |
+---+----+---+      +---+----+---+
| IoT node 1 |      | IoT node 2 |
+------------+      +------------+
```

Fig. 2 IoT Data Collection

3. Terminologies

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

Receiver: is the target for the data publication.

Publisher: pushes data to the receiver according to the subscription information.

Subscription Server: which manages capabilities that it can provide to the subscriber.

Global Subscription: the subscription requested by the subscriber. It may be decomposed into multiple Component Subscriptions.

Component Subscription: is the subscription that defines the data from each individual telemetry source which is managed and controlled by a single Subscription Server.

Global Capability: is the overall subscription capability that the group of Publishers can expose to the Subscriber.

Component Capability: is the subscription capability that each Publisher can expose to the Subscriber.

Master Publication Channel: the session between the Master Publisher and the Receiver.

Agent Publication Channel: the session between the Agent Publisher and the Receiver.

4. Solution Overview

All the use cases described in the previous section are very similar on the data subscription and publication mode, hence can be abstracted to the following generic distributed data collection framework, as shown in the following figure.

A Collector usually includes two components,

- 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.

```
     +-------------------------------------+
     |      Collector      |-------------+ |
     |                    +------------+ | |
     |   +------------+   || Receiver  | | |
     |   | Subscriber |   |--------------+ |
     |   +-^----+-----+   +---^--------^   |
     |     |    |             |        |   |
     +-------------------------------------+
Global     |    |Global       |Push    |
Capability |    |Subscription |        |
     +------------------------+-----+  |
     |     |    | Publisher(Master) |  |
     |  +--+----v------+            |  |
     |  | Subscription |            |  |
     |  | Server       |            |  |
     |  +--^----+------+            |  |
     |     |    |                   |  |
     +------------------------------+  |
Component  |    | Component            |Push
Capability |    | Subscription         |
     +------------------------------+  |
     |     |    | Publisher(Agent)  |  |
     |  +--+----v------+            |  |
     |  | Component    |            |  |
     |  | Subscription |            +--+
     |  | Server       |            |
     |  +--------------+            |
     +------------------------------+
```

Fig. 3 The Generic Distributed Data Collection Framework

Master and Agents may interact with each other in several ways:

- o Agents need to have a registration or announcement handshake with the Master, so the Master is aware of them and of life-cycle events (such as Agent appearing and disappearing).

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

- o The Master relays the component subscriptions to the Agents.

- o The Agents indicate status of Component Subscriptions to the Master. The status of the overall subscription is maintained by the Master. The Master is also responsible for notifying the subscriber in case of any problems of Component Subscriptions.

Any technical mechanisms or protocols used for the coordination of operational information between Master and Agent is out-of-scope of the solution. We will need to instrument the results of this coordination on the Master Node.

5. Subscription Decomposition

Since Agents are invisible to the Collector, the Collector can only subscribe to the Master. This requires the Master to:

1. expose the Global Capability that can be served by multiple Publishers;

2. disassemble the Global Subscription to multiple Component Subscriptions, and distribute them to the corresponding telemetry sources;

3. notify on changes when portions of a subscription moving between different Agents over time.

To achieve the above requirements, the Master need a Global Capability description which is typically the YANG [RFC7950] data model. This global YANG model is provided as the contract between the Master and the Collector. Each Agent associating with the Master owns a local YANG model to describe the Component Capabilities which it can serve as part of the Global Capability. All the Agents need to know the namespace associated with the Master.

The Master also need a data structure, typically a Resource-Location Table, to keep track of the mapping between the resource and the corresponding location of the Subscription Server which commits to serve the data. When a Global Subscription request arrives, the Master will firstly extract the filter information from the request. Consequently, according to the Resource-Location Table, the Global Subscription can be disassembled into multiple Component Subscriptions, and the corresponding location can be associated.

The decision whether to decompose a Global Subscription into multiple Component Subscriptions rests with the Resource-Location Table. A Master can decide to not decompose a Global Subscription at all and push a single stream to the receiver, because the location information indicates the Global Subscription can be served locally by the Master. Similarly, it can decide to entirely decompose a Global Subscription into multiple Component Subscriptions that each push their own streams, but not from the Master. It can also decide to decompose the Global Subscription into several Component Subscriptions and retain some aspects of the Global Subscription itself, also pushing its own stream.

Component Subscriptions belonging to the same Global Subscription MUST NOT overlap. The combination of all Component Subscriptions MUST cover the same range of nodes as the Global Subscription. Also, the same subscription settings apply to each Component Subscription, i.e., the same receivers, the same time periods, the same encodings are applied to each Component Subscription per the settings of the Global Subscription.

Each Component Subscription in effect constitutes a full-fledged subscription, with the following constraints:

- o Component subscriptions are system-controlled, i.e. managed by the Master, not by the subscriber.

- o Component subscription settings such as time periods, dampening periods, encodings, receivers adopt the settings of their Global Subscription.

- o The life-cycle of the Component Subscription is tied to the life-cycle of the Global Subscription. Specifically, terminating/ removing the Global Subscription results in termination/removal of Component Subscriptions.

- o The Component Subscriptions share the same Subscription ID as the Global Subscription.

6. Publication Composition

The Publisher collects data and encapsulates the packets per the Component Subscription. There are several potential encodings, including XML, JSON, CBOR and GPB. The format and structure of the data records are defined by the YANG schema, so that the composition at the Receiver can benefit from the structured and hierarchical data instance.

The Receiver is able to assemble many pieces of data associated with one subscription, and can also deduce the missing pieces of data. The Receiver recognizes data records associated with one subscription according the Subscription ID. Data records generated per one subscription are assigned with the same Subscription ID.

For the time series data stream, records are produced periodically from each stream originator. The message arrival time varies because of the distributed nature of the publication. The Receiver assembles data generated at the same time period based on the recording time consisted in each data record. In this case, time synchronization is required for all the Publishers.

To check the integrity of the data generated from different Publishers at the same time period, the Message Generator ID [I-D.ietf-netconf-notification-messages]is helpful. This requires the Subscriber to know the number of Component Subscriptions which the Global Subscription is decomposed to. For the dynamic subscription, the reponse of the "establish-subscription" and "modify-subscription" RPC defined in [I-D.ietf-netconf-subscribed-notifications] can 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] can also include a list of Message Generator IDs to notify the current Publishers for the corresponding Global Subscription.

7. Subscription State Change Notifications

In addition to sending event records to receivers, the Master MUST also send subscription state change notifications[I-D.ietf-netconf-subscribed-notifications] when events related to subscription management have occurred. All the subscription state change notifications MUST be delivered by the Master Publication Channel which is the session between the Master Publisher and the Receiver.

When the subscription decomposition result changed, the "subscription-modified" notification will be sent to indicate the new a list of Publishers.

8. IANA Considerations

TBD

9. Security Considerations

It’s expected to reuse the existing secure transport layer protocols, such as TLS [RFC5246] and DTLS [RFC6347], to secure the telemetry stream.

10. Acknowledgements

TBD

11. References

11.1. Normative References

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

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

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

11.2. Informative References

[I-D.ietf-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", draft-ietf-core-coap-pubsub-05 (work in progress), July 2018.

[I-D.ietf-netconf-notification-messages]
Voit, E., Birkholz, H., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-04 (work in progress), August 2018.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher’s Event Streams", draft-ietf-netconf-subscribed-notifications-17 (work in progress), September 2018.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-19 (work in progress), September 2018.

Appendix A. Change Log

(To be removed by RFC editor prior to publication)

v01

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

v02

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

v03

- o Added a section for Terminologies.
- o Added a section for Subscription State Change Notifications.
- o Improved the Publication Composition section by adding a methed to check the integrity of the data generated from different Publishers at the same time period.
- o Revised the solution overview for a more clear description.

Authors' Addresses

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Eric Voit
Cisco Systems
United States of America

Email: evoit@cisco.com

Alexander Clemm
Huawei
2330 Central Expressway
Santa Clara, California
United States of America

Email: alexander.clemm@huawei.com

Andy Bierman
YumaWorks
United States of America

Email: andy@yumaworks.com